PERSONALIZATION OF UNIVERSAL SMART CARD MICROCONTROLLER FOR SECURE MANAGEMENT SYSTEMS

BY
CSUO AAMIR JAVED (GP LDR)
GC ZAHEER AHMED
CAPT IFTIKHAR
PC WAQAS

Submitted to the Faculty of Computer Science Department,
Military College of Signals, National University of Sciences and Technology,
Rawalpindi in partial fulfillment for the requirements of a B.E. Degree in Computer Software Engineering

MAY, 2005

ABSTRACT

Personalization of Universal Smart Card Microcontroller for Secure Management Systems

by

CSUO Aamir Javed

GC Zaheer Ahmed

Capt Iftikhar

PC Waqas

Cards with magnetic strips are used all over the world for authentication between card and service provider and in the case of a signature or a PIN code between individuals and the service provider. Though their security is not strong they are accepted by the user. In the case of authentication for the first time a smart card was introduced for telecommunication purposes. This card enables a strong authentication between user and service provider. So we are going to use this authentication scheme in developing secure management systems.

This project involves the design and implementation of a universal smart card framework used to replace the growing number of cards each of us carries around. The smart card will have three main features; to allow for easier transportation of secure data, allow authorization to a number of different devices and finally to allow the positive identification of the owner. Applications will also be created during this project to demonstrate the functions of the smart card, such as the Smart Card based Time and Attendance Management System, Smart Card Based Library Management System& Smart Card Access Control System. A number of issues will be focused upon; the main will be security of any personal information stored on the card, but another will be making the card flexible so it can adapt with its changing environment.

DEDICATION

To our Parents and our Instructors

DECLARATION

No portion of the work presented in this dissertation has been submitted in support of award of qualification either at this institute or elsewhere.

ACKNOWLEDGMENTS

All praise is to ALMIGHTY ALLAH who bestowed us with this much of knowledge that we are able to accomplish this great challenge fruitfully.Special thanks to Our Project Supervisor Lec Aihab for his cooperation and guidance in every field. We are also thankful to those people who have provided their kind assistance in achieving this task.

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

LITERATURE REVIEW

Smart cards are a part of everyday lives, and as days go by, smart cards are becoming increasingly popular. The credit card size smart card, embedded with a chip that stores memory, is definitely not anything new. Smart cards have been around for a while, but still they are not too popular. Smart cards are at work in more than 90 countries; however, many issues associated with the smart card hinder its acceptance among people. The smart card technology offers a lot of new features and has a lot of positive points, but there are also negative points[1].

Many countries, states, and even organizations are starting to use smart card technology. One major concern about the smart card is whether it will violate the privacy of individuals. Another related concern is the security available protecting the information stored on smart cards. Although the privacy and security issues are prevalent among United States' citizens, many people welcomed the use of smart cards after the 9/11 terrorist attacks[2]. Questions are beginning to emerge about the threat of smart card technology against citizen's rights.

There is a much more efficient way to have someone sign something than the "old" technique of mailing and processing forms manually. Utilizing a digital signature or an e-signature to sign an online form is much more effective and less time consuming .For the simple fact that smart cards are difficult to duplicate, businesses could use them during their times of network log-in and procedures used for validation.

Despite the fact that technology is thriving and many people anxiously anticipate any new advances to make their lives compliant in order for the smart card to become popular, they need to be less costly. When purchasing or investing in a product, the cost plays a substantial part in the buyer's decisiveness. Card readers constructed into computers, mobile phones, and other hand-held devices would be much more convenient than a card reader looking like an ATM machine. Since the smart card business is still not booming, extensive research and trials will have to be executed in order to fully implement the technology. Therefore, the introductory card trials will need various cards for numerous tasks that will hopefully later be able to be fulfilled from a single card.

Cards will need to be maturated to become very consumer-friendly. These cards will be a first-rate business tool and make the lives of many people a lot easier and less chaotic once their use is sanctioned[3].

It is crucial to explore the effectiveness of smart card technology in offering security framework guaranteeing protection of individual privacy. Smart cards could one day be a part of everyday lives, accruing, maintaining, and manipulating tremendous amounts of data in various options such as health care, national identification, e-government programs, domestic and international travel, among many other things.

There are many different smart card applications. A few examples are national identification, financial transactions, e-government programs, information security, physical access, travel data, retail and loyalty, health records, and university activities.National identification has been one of the foremost uses of smart cards in many countries.

Financial applications deal with things such as e-cash and electronic purses. E-cash allows people to put a certain amount of money on the card and engage in transactions electronically at the, bank, and many other places. An electronic purse is similar to e-cash, but it used in the place of change. For example, one might buy a drink using a smart card, instead of having to come up with the cash. Also, credit and debit accounts are incorporated into this application allowing the card to offer secure use via the Internet. Also, the mass transit collection and electronic toll collection systems can be incorporated into this application. Instead of having to pay a toll every morning at the bus station, the toll would be paid with a swipe of a magnetic stripe card or the wave of a contactless card.

The communications application deals with things such as calls, mobile communication, and pay-tv. This application allows the user to make long distance phone calls and order a pay-per-view movie.

The e-government programs application involves using the card for things such as WIC (Women Infants and Children) or food stamps. WIC is usually issued through vouchers. Also, at one time food stamps were issued through paper. However, with this application WIC and food stamps can be disbursed through an electronic means. Also, one day voting among other things could all be handled online. Or voting could be done using a card that had individuals' information such as party affiliation. Many of the things that are done by or using paper, would become paperless and began to be electronic[6].

The information security and physical access applications are used for employee access to computer systems and restricted facilities. Also, they are used to give open secure areas. This application could be very effective in keeping unwanted people from gaining access to certain places. The transportation application incorporates many things. One of the biggest things is the driver's license. The health card application contains insurance data, emergency medical data, and medical records. With a swipe of a card, one could know the blood type of an individual requiring emergency medical treatment.

Retailers use the retail and loyalty application to tract down their faithful customers. Consumers would then be given coupons or incentives to spend more. So if a customer spent $100 at a store, they could receive a $10 gift certificate in the mail. This application also allows retailers to monitor the spending habits of their customers and reward them accordingly. Sometimes, however, lawyers have used this accumulated data to use against customers.

Lastly, there is the University application. This application combines a meal plan, an electronic purse, and a library card, among other things, all on one card. A student could was clothes, check out a book, and eat dinner, all with the same card. All of the previous applications could include and aspect called biometric technology, which includes things such as fingerprint and face recognition. This technology would make it virtually impossible or at least perplexing for someone to hack into the application.

This project involves the design and implementation of a universal smart card framework used to replace the growing number of cards each of user carries around.

The smart card will have three main features; ***to allow for easier transportation of secure data, allow authorization to a number of different devices and finally to allow the positive identification of the owner.*** Applications will also be created during this project to demonstrate the functions of the smart card, such as the Smart Card based Time and Attendance Management System & Smart Card Access Control System. A number of issues will be focused upon; the main will be security of any personal information stored on the card, but another will be making the card flexible so it can adapt with its changing environment.

CHAPTER 2

INTRODUCTION TO SMART CARD

2.1 Introduction

Every so often, a technology emerges with the potential to affect virtually every part of daily life. Smart cards - plastic cards housing a “smart” silicon chip with the power to store and process information - are such a technology. Throughout the world, smart cards are being used in an ever-increasing variety of ways - for numerous banking and Internet payment projects, for ticketing in mass transit systems, for social service provision, and to enable digital mobile communications.

The silicon chips inside smart cards do three things. Firstly, and essentially, they store information, for which they use a nonvolatile memory. This is a memory that retains its content even when it is not powered up. Secondly, they protect the information. Protection can be by an access code such as a four-digit PIN number, or by encrypting the information to make it difficult for intruders to read. For this the chips often include a microcontroller. Thirdly, the chip communicates with the outside world. In order for them to be used by a number of different card readers, these means of communication (interfaces) are standardized [1].

ISO uses the term, Integrated Circuit Card (ICC) to encompass all those devices where an integrated circuit is contained within an ISO 1 identification card piece of plastic as shown in Figure 2:1.

The card is 85.6mm x 53.98mm x 0.76mm and is the same as the ubiquitous bank card with its magnetic stripe that is used as the payment instrument for numerous financial schemes.

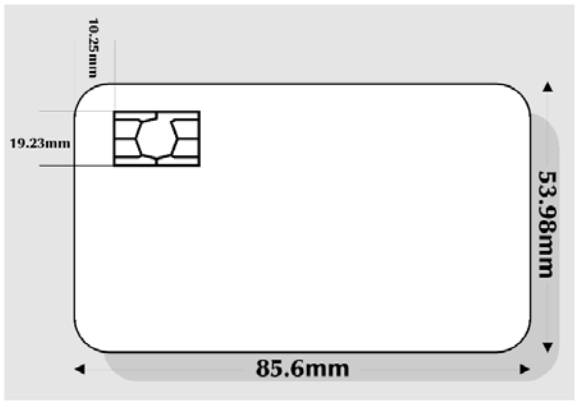


Figure 2:1 ISO ID 1Card

2.2 Smart Card vs. Magnetic Stripe Card

The primary difference between a smart card and a magnetic stripe card is that a smart card stores information or monetary value on a computer chip, rather than on a magnetic stripe. The magnetic stripe card system is currently the most popular card. However, it was designed for the technological infrastructure that existed in the 1960s. The smart card system is a new system that is still being adopted. The smart card system is much more secure and fits in with the technological infrastructure of today [2].

The most widely known example of magnetic stripe card is the credit card. The only information that is encoded onto the magnetic stripe is information that identifies the cardholder and the card issuer. The rest of the information about the cardholder, as well as data, account options, etc. is designed to be held remotely on large, central mainframe computers. In order for the magnetic stripe card system to function, it is dependent upon large, remote computer systems and dedicated communications networks. Therefore, magnetic stripe cards can only be used at locations that are connected to dedicate networks [3].

In magnetic-stripe technology, a strip of tape with a surface layer of magnetic material is magnetized in specific patterns that represent the encoded data. When the card is passed through a reader, the magnetized bits generate electric impulses whose patterns are interpreted as specific characters. A magnetic stripe card usually has three tracks of data. Two tracks are designed as read-only tracks, while the third is used as a read-and-write track to monitor daily use in an offline environment.

There are two main reasons why magnetic-stripe technology is becoming more and more insufficient. First of all, the low data-storage capability on the card makes it an impractical storage device for offline use in complex applications. Secondly, it very vulnerable to fraudulent use. The increasing use of magnetic-stripe cards for consumer-finance transactions has created a massive fraud opportunity.

2.3 Types of Smart Cards

The two groups of smart cards include Contact and Contactless smart cards[5].

2.3.1 Contact Smart Cards

Contact smart cards must be inserted and swiped through a reader slot in a Point of Service (POS) device. Contact cards use a series of metal pins that touch metal plates on the card surface to complete an electrical circuit. The Contact Card is the most commonly seen ICC to date largely because of its use in France and now other parts of Europe as a telephone prepayment card.. Most contact cards contain a simple integrated circuit although various experiments have taken place using two chips. The chip itself varies considerably between different manufacturers and for a whole gambit of applications.

2.3.2 Contactless Smart Cards

Contactless smart cards do not need to be inserted and swiped through a reader slot. They employ radio frequencies to communicate between the cards and readers (an electrical field is used to induce a current in a series of coils embedded within the plastic of the card). The Contactless card may contain its own battery, particularly in the case of a "Super Smart Card" which has an integrated keyboard and LCD display. In general however the operating power is supplied to the contact less card electronics by an inductive loop using low frequency electronic magnetic radiation. The communications signal may be transmitted in a similar way or can use capacitive coupling or even an optical connection.

2.3.3 Comparison

The chip capabilities of the two cards are basically identical. However, due to lower volume and simpler technology requirements of typical Contactless applications, a greater degree of diversity of memory specifications and faster adoption of improved technical processes are occurring in the Contact smart card environment.

2.4 Smart Card Contacts

Smart card contacts are the channel through which card connects to the card reader. The card require normally 5V operating voltage provided to it through card reader. The smart card external contacts are VCC supply voltage, RST is to reset the card, CLK is the frequency provided to the card CPU,VPP is 5V programming voltage, I\O is the channel through which card communicates with reader for data. Smart card contacts are defined in table 2:1 and shown in Figure 2:2

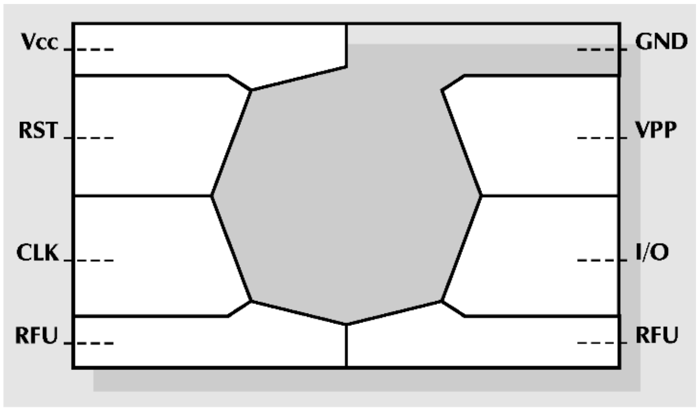


Figure 2:2 Smart Card Contacts

Smart card contacts are defined in table 2:1

Table2:1 Smart Card Contacts

CONTACT	**DESIGNATION**	**USE**
C1	VCC	Power connection through which operating power is supplied to the microprocessor chip in the card
C2	RST	Reset line through which the IFD can signal to the smart card's microprocessor chip to initiate its reset sequence of instructions
C3	CLK	Clock signal line t hrough which a clock signal can be provided to the microprocessor chip. This line controls the operation speed and provides a common framework for data communication between the IFD and the ICC
C4	RFU	Reserved for future use
C5	GND	Ground line providing common electrical ground between the IFD and the ICC
C6	VPP	Programming power connection used to program EEPROM of first generation ICCs.
C7	I/O	Input/output line that provides a half-duplex communication channel between the reader and the smart card
C8	RFU	Reserved for future use

2.5 Smart Card Chip

Smart cards have a memory architecture that will be unfamiliar, if not downright bizarre, to most mainstream programmers. Programmers typically think in terms of having available large amounts of homogeneous random access memory (RAM) that is freely available for reading and writing. This is definitely not the case on a smart card. There are, in fact, three kinds of memory on a smart card: read-only memory (ROM), nonvolatile memory (NVM), and a relatively tiny amount of random access memory (RAM). The central processing unit in a smart card chip is an 8-bit microcontroller, typically using the Motorola 6805 or Intel 8051 instruction set. Hitachi’s H8 smart card chip is a notable exception. These instruction sets have the usual complement of memory and register manipulations, addressing modes, and input/output operations[7].

Some chip manufacturers have extended these basic instruction sets with additional instructions that are of particular use on smart cards. Smart card CPUs execute machine instructions at the rate of about 400,000 instructions per second (400

KIP), although speeds of up to 1 million instructions per second (1 MIP) are becoming available on the latest chips. This Smart Card Chip Architecture is shown in Figure 2:3. The processor has four peripherals:

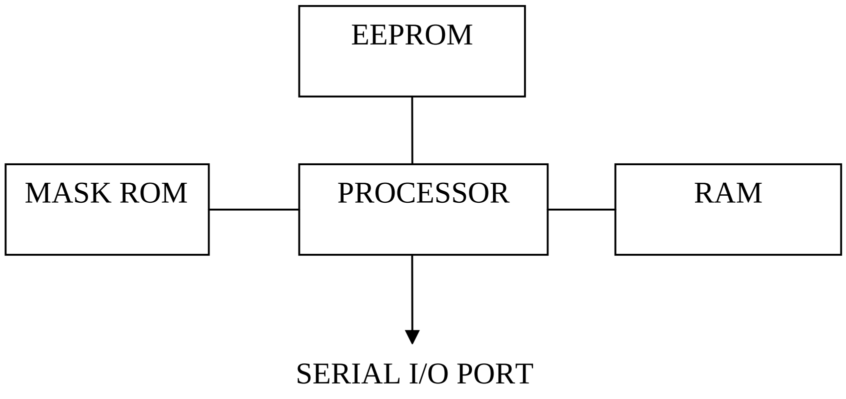


Figure 2:3 Smart Card Chip

2.5.1 Mask Rom

The mask ROM contains the operating system of the chip and is made as part of the chip fabrication process. This memory is read only and cannot be changed once the chip is made. The ROM may contain programs and data but in both cases the code and data are constant for all time. By the very process that the chips are made it is not practical to have any form of unique code or data in ROM. Thus the ROM memory is constant for a batch of chips (thousands). Each wafer at the end of the manufacturing process results in the die (apart from fabrication failures) looking identical.

2.5.2 EEPROM

The EEPROM memory is the non-volatile storage area of the chip that allows data to be written and read under program control. This data is preserved even after the power to the chip is switched off. By writing data into the EEPROM a unique identity can be given to each chip. The Smart Card chips from most semiconductor manufacturers have the facility to make parts of the EEPROM memory `write once only'. This is sometimes called OTP (One Time Programmable) or occasionally as EPROM memory in the sense that it cannot be overwritten. The latter term is ambiguous in that although EPROM memory requires ultra violet light for erasure, in the general sense the memory cells are always capable of being set to the final state.

Thus if the initial state is all `ones' then any bit can be overwritten to `zero'. If this situation is allowed to arise then in some circumstances you may be subject to a security violation. Under these conditions going from a `1' to a `0' must increase the security for every bit used. A reverse situation may allow an attacker to decrease the security by over writing a `1' to a `0' which is an inherently possible process[10].

2.5.3 RAM

The random access memory (RAM) forms the memory working space to be used by the processor whilst executing programs either in ROM or EEPROM. This memory is volatile and all data will be lost when the power to the chip is removed.

2.5.4 Serial I/O Port

The serial I/O port should be considered as just another peripheral to the processor which may be read and written under software control. The most important point to notice is that the hardware sophistication often found on general purpose microprocessors has been removed to optimize the space available on the silicon. Thus the ubiquitous UART (Universal Asynchronous Receiver Transmission) which buffers bytes of data to and from the serial port is replaced by a single register that the programmer must manage on a bit by bit basis. Further more the timing of data transmission which is handled by the UART must now be managed by the program in the Smart Card [11].

2.6 Chip Operating System

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

The general purpose COS which features a generic command set in which the various sequences cover most applications, and the dedicated COS with commands designed for specific applications and which can even contain the application itself.

An example of a dedicated COS would be a card designed to specifically support an electronic purse application.

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

2.7 Chip Cycle

The manufacturing of smart cards comprises a number of distinct steps: Fabrication of the chip, or many chips in the form of a wafer, Packaging of individual chips for insertion into a card, Fabrication of the card, Pre-personalization, Personalization, Printing of the card, and Initialization of the program and program information on the chip in the card. There are of course many variations for the IC card life cycle depending on the requirement of the particular application.

Applications that make use of smart cards typically comprise software that runs on a reader-side computer (a PC or PC class computer), on the smart card itself, and perhaps even on other computers widely distributed within a local area or wide area network. Development of such applications require special considerations to fully incorporate the smart card, both with respect to development of software to go on the card itself and in testing the smart card in the full application. Figure 2:4 shows the basic life cycle of chip.

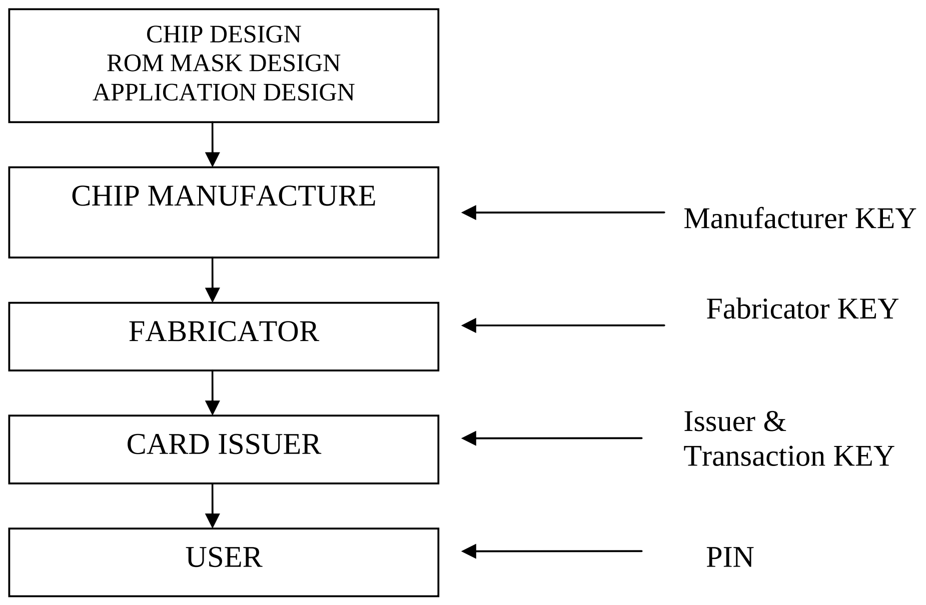


Figure 2:4 Chip Lifecycle

2.8 Smart Card memories

The Smart Card chip is often referred as a device containing a Microcontroller and memory[14]. This really belies the story behind some very sophisticated technology and in particular the developments of advanced memory techniques. Division of memory is shown in Figure 2:5

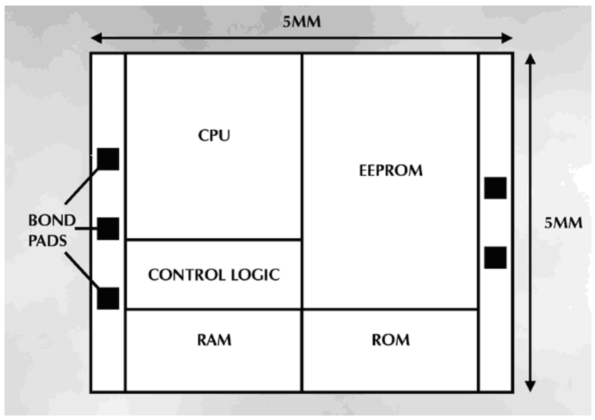


Figure 2:5 Smart Card memories

The more commonly used memory types are: ROM(Read only memory (mask ROM)),PROM (Programmable read only memory),EPROM(Erasable programmable ROM),EEPROM(Electrically erasable PROM),RAM(Random access memory).

ICC can be differentiated by the different types of their content, Memory only, Memory with security logic, Memory with CPU. The security logic can be used to control access to the memory for authorized use only. This is usually accomplished by some form of access code which may be quite large (64 bits or more). Clearly the use of EEPROM memory must be strictly controlled where fraudsters can obtain a financial advantage by unauthorized use. This applies as much to telephone cards as applications using ICC for cryptographic key carriers. The security advantage of the CPU device is of course more significant because the CPU is capable of implementing cryptographic algorithms in its own right.

2.9 Smart Card T=0 Protocol

The T=0 protocol is a byte-oriented(Byte-oriented means that a byte is the unit of information transferred across a channel and that error handling is handled one byte at a time as well.) Protocol. Like all other ISO-compliant smart card protocols, it functions in a command-response mode in which the reader-side of the connection issues a command to the card, which then performs the commanded operation and sends back a response[10].

In the T=0 protocol, error detection is done by looking at the (even) parity bit on each byte transferred across the reader-to-card interface. The transfer of each byte of information requires the use of 10 bits, as illustrated in Figure 2:6. The parity bit is cleared or set to make the total number of bits set (per character transferred) be an even number. The receiver side of the channel can look at the bit values transferred prior to the parity bit and determine whether the parity bit should be set. If the actual parity bit transferred does not match what was expected, then it can be assumed that an error exists in the byte of data just transferred and some recovery procedure must be undertaken. The recovery procedure used with the T=0 protocol is triggered by the receiving side, which, on detecting a parity error, signals that it expects the transmitting side to retransmit the byte (that was received in error). It provides the signal to the transmitting side by holding the I/O line in a low state. Normally, the I/O line is in a high state immediately preceding the transfer of a byte, so a low state acts as an error feedback signal to the transmitter. On detecting this, the transmitting side of the channel waits for at least two character times and then again sends the byte that was previously received in error as shown in Figure 2:6.

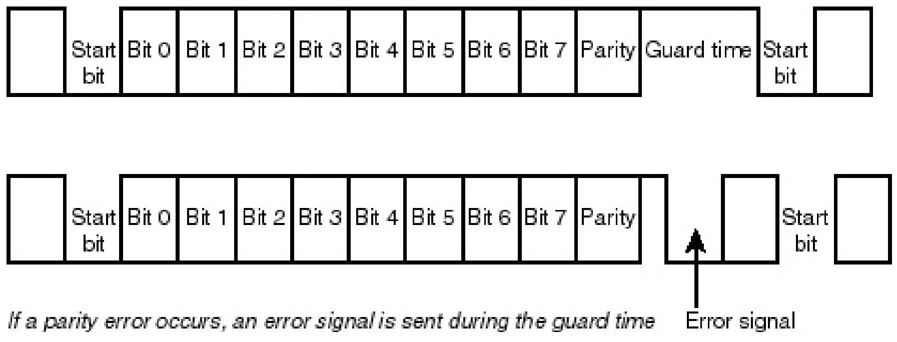


Figure 2:6 Smart Card T=0 Protocol

The application protocols exchange information through data structures referred to as application protocol data units (APDUs).The APDU for the T=0 protocol comprises two distinct data structures: one that is sent from the reader to the card (as a command) and one that is sent from the card to the reader (as a response) as shown in Figure 2:7.

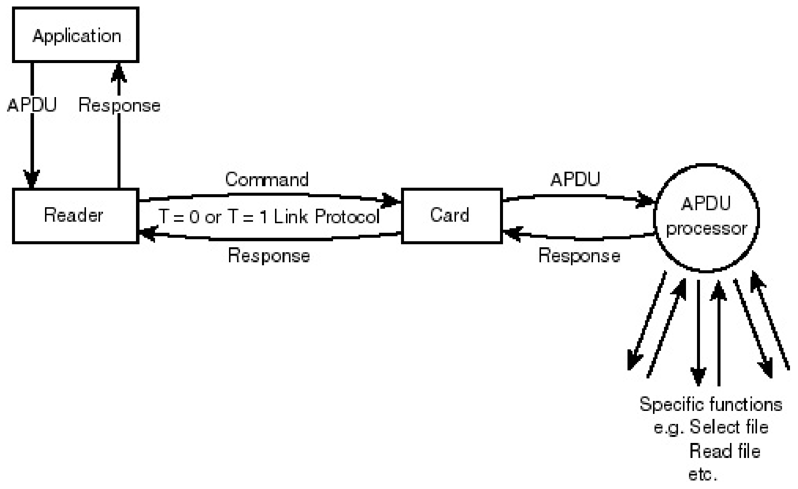


Figure 2:7 Smart Card T=0 Protocol

2.10 Multi Application Smart Cards

Multi application referred to a tool that could implement different operations for different clients. The multi function card has a defined single owner who can set the commercial rules for use of the card and define the necessary branding issues. Most of the more complex applications to date have been along these lines in that the functions have all been defined by one provider or there has been a substantial cooperation to share the management of the Smart Card.It is the multi application card that is really the more interesting. In this project has a Smart Card that is no more than an application carrier for the various application providers. These providers ideally should be totally unrelated [18] as shown in Figure 2:8.

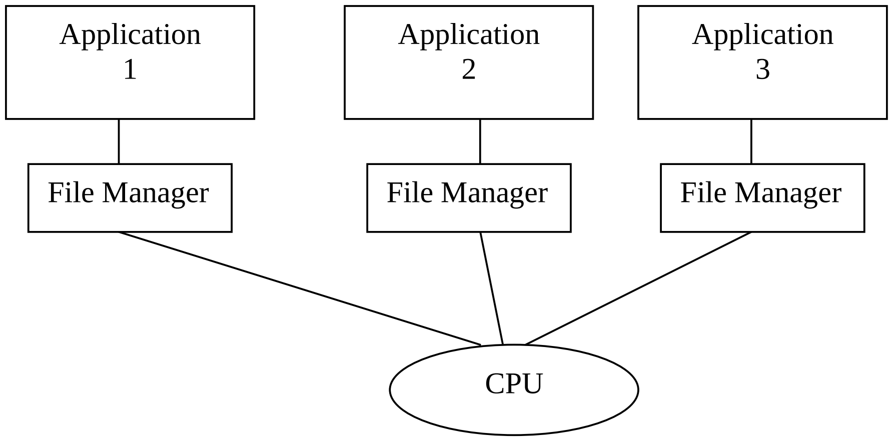


Figure 2:8 A Generalized Application

CHAPTER 3

SECURITY AND THE SMART CARD

3.1 Smart Card System Components

3.1.1 The Card

Smart cards use a computer platform on which information can be stored such that access to it can be strictly controlled by the cardholder, the card issuer, or the provider of any specific applications on the card. Further, software can be executed on the card under strict control of the cardholder, the card issuer, or the provider of specific applications on the card. Given these characteristics, the smart card provides a variety of useful security characteristics, including Storage of passwords for access to computer systems, networks, information stores, and so on. Storage of keys, public and private, for authenticating identity[15].

Storage of keys, public and private, for encrypting information to ensure its privacy. Storage of information to be conveyed to various access points for a system (for example, a financial system) without the cardholder being able to access or change that information in any way ,Performance of encryption algorithms for authenticating identity, Performance of encryption algorithms for ensuring the privacy of information.

3.1.2 The Cardholder

A smart card can represent the cardholder in an electronic environment. Further, the card can be programmed to require some type of identity authentication from the cardholder before it will provide such electronic representation for the cardholder. That is, the smart card can use a variety of mechanisms in a transaction with the cardholder through which the cardholder convinces the card that it should act on the cardholder's behalf. Some of the mechanisms used by the card to authenticate the identity of the bearer include Requiring the bearer to enter a personal identification number (PIN), Requiring the bearer to enter some known personal information stored on the card, Requiring some biometrical characteristic of the bearer, such as a fingerprint or a facial image, to be measured by a sensor or collection of sensors and then matched against a benchmark of this characteristic stored on the card.. Requiring the bearer to properly perform a series of operations leading to a specific state known to the card[16].

The identity authentication transaction that occurs between the card and the cardholder is a rather complete specific example of the transaction that one wants to occur generally through the enabling actions of the card. Both sides of the transaction (that is, the card and the cardholder) must be concerned with authenticating the identity of the other (party to the transaction). Being authorized with the appropriate privileges once identity is authenticated .Being assured of the integrity of the transaction Being assured of the privacy of the transaction. Being able to confirm that the transaction took place in a proper fashion.

3.1.3 The Card Issuer

The card is typically given to the cardholder by a card issuer. In the case of financial cards, the issuer is generally a bank or other financial institution. The card issuer generally is responsible for providing the system in which the card can function to perform its security-related functions. One aspect of this system is typically the linking of salient information about the cardholder to the functional characteristics of the card. The issuer functions as a certification authority or as a trust broker. It is through the actions of the issuer that various parties of a subsequent transaction can achieve some level of trust in the transaction, although they do not know each other prior to the initiation of the transaction[13].

In the financial environment, very well-defined protocols have been put in place by associations of financial organizations and buttressed by binding national and international laws and agreements. In the emerging world of computer networks, the existence of equivalent certification authorities is only now being legitimized by evolving system deployment.

3.1.4 The Terminal

The access point of any smart card with any electronic system is typically referred to as a terminal; sometimes the terms smart card reader or smart card interface device are also used. Terminals can vary significantly in complexity and capability and hence in the level of security that they support. At the most capable level, a terminal is a secure computing platform on par with the smart card itself, although typically not nearly so small, inexpensive, and portable. In such a configuration, a terminal might contain a comparatively powerful computer processor, memory, telecommunications interfaces to local and wide area computer networks, display screens, input devices (for example, a keypad or keyboard) through which a user can enter information (to the terminal's processor and then perhaps on to the smart card), and perhaps even biometric sensors that the terminal can use to ascertain personal characteristics of the cardholder. For example, fingerprint readers and facial characteristics scanners are beginning to emerge within the security marketplace as viable elements of terminals.

A highly integrated configuration including a tamper-resistant computer, memory and secondary storage, and a secure cardholder verification entry facility would typically be provided by a card issuer to a merchant. This terminal provides a secure point of presence (from the standpoint of the card system issuer) in the merchant's environment through which the issuer system can communicate with the cardholder's smart card [19].

3.1.5 The PC

In emerging computer network environments, the terminal component from earlier smart card-based systems is separated into a computer component (that is, a PC, a network computer, a workstation, or some similar designation) to which is attached a relatively simple smart card reader. This particular configuration raises some security concerns with respect to the use of smart cards. In particular, the cardholder should always understand the security risks in providing verification of identity to the card through a computer system of unknown control. Obviously, this same concern can be raised for all levels of systems; Trojan horse ATMs reportedly have been deployed to fraudulently gain account numbers and PINs from unsuspecting users. For home computer-type systems, however, the risks of the system being in a position to capture sensitive information are significantly greater.

The point then is that the cardholder should be reasonably cautious of the computer systems through which the card is used. If it's the personal computer system of the cardholder, then the risks are greatly minimized since the cardholder has control over the system's security environment. If it's a personal computer system being used in a commercial environment, then the cardholder should be concerned with the manner in which a PIN is entered.

For example, if a personal computer configuration makes use of a simple smart card reader and the cardholder is expected to enter a PIN through the computer's keyboard, then it's a relatively simple procedure for the system manager of the personal computer configuration to be able to capture the keystrokes and know the PIN for the cardholder's card. If the computer belongs to the cardholder and is under the direct control of the cardholder, then the security risks (of having the PIN captured) are greatly minimized. For public environments, it is possible to obtain more sophisticated smart card readers which have integrated keypads through which a PIN can be entered and passed on to the card, not to the computer system to which this terminal is connected.

3.1.6 The Network

The network through which computer systems are connected should always be treated as a completely no secure environment. The application developer, the card issuer, and the cardholder should all view the communication channel as completely open to the world. Information that passes through these channels can be monitored, captured, and manipulated by unknown persons or systems.

3.1.7 The Application

The application is the particular system or system component that is provided through the auspices of the card issuer (or at least with the concurrence of the card issuer) and is intended to provide some type of service accessed by the cardholder. The application may make use of an infrastructure within the network or within the

end computer system through which the cardholder gains access to it. In these cases, however, the application must be concerned with the security of this infrastructure.
In many existing smart card-enabled systems, all the players operate within an environment provided by the card issuer. In the Internet environment, it is more difficult to provide a well-controlled infrastructure for all these players. They must each understand the security limits of the components that they deal with.

3.2 Smart Card System Design

Most smart card programming consists of writing programs on a host computer that send commands to and receive results from predefined or application-specific smart cards. These applications read data from and write data to the smart card and perhaps make use of the modest computing powers of the processor on the smart card. Smart cards in these applications are typically secure stores for data pertaining to the individual bringing the card to the system, such as personal identification data.

In situations where no off-the-shelf card contains all the functionality needed by the application, the programmer may be able to extend the capabilities of an off-the-shelf card by writing software that runs on the card itself. This software may implement special-purpose or higher-level functions on the card that is a combination of existing operating system functions, or it may provide additional protections for the data stored on the card.

Finally, there may be situations where the operating system capabilities of an existing smart card need to be extended, or where a new and unique smart card needs to be manufactured. Examples of such situations include a closed system application where cost or a particularly high level of security is a critical factor, or where a particular encryption algorithm is needed to connect the smart card to an existing host system. In these situations, smart card programmers write new operating system software for smart cards partially or completely in the assembly language of the processor on the smart card[20].

Regardless of the type of software being written, the smart card programmer must be constantly aware of the two central concerns of smart card software: data security and data integrity.

3.3 Data Security

Data security means simply providing data only to those who are authorized to receive it. It requires that neither data values nor even information about these data values are revealed to unauthorized parties or systems. Wherever data is stored and whenever data is moved, the smart card programmer must ensure that this requirement is satisfied.

Although there are many methods for obscuring data, data security doesn't make use of data encryption as much as it does the notion of an authorized entity. One of the most obvious attacks on a smart card is for an unauthorized entity to become authenticated as an authorized entity, then use the provided facilities of the card to access data. Because keys, passwords, and PINs stored on the card are all used to authenticate entities, particular care must be exercised in protecting this kind of data.

One way to ensure the data security of a smart card is to control physical access to the card. This technique is often used early in the life cycle of the card. For example, cards are manufactured under tight physical security procedures and shipped to the card customer under equally tight procedures. The keys used to protect the card during transit from manufacturer to customer are called transport keys and they are given only to the customer. Using the transport keys, a smart card customer can access all files on the smart card and set up a particular file system and access authorization scheme. Transport keys are like a super user password for the card and are typically deleted from the card by the customer after they have been used to configure the card for the customer's application.

3.4 Data Integrity

Data integrity requires that all parties to a smart card transaction agree on the state of the data in the transaction. If vending machine software intends to charge a smart card electronic purse $1 for a can of soda, then at the end of the transaction, there should be $1 less on the smart card, $1 more in the vending machine, and a soda in the hands of the cardholder. Any variation—$1 added but $1 not subtracted, $1 transferred, but no soda vended, and so on—is a violation of data integrity.

The primary source of breakdowns in data integrity are system failures in the middle of processing a transaction. Such failures can be accidental—for example, when a communication line from an ATM goes down—or deliberate—for example, when the cardholder prematurely removes the smart card from the reader. Programming to ensure data integrity means both guarding against its loss and detecting and repairing its loss when it occurs.

3.5 Security requirements

3.5.1 Physical Security

Central to the overall security architecture is the concept of physical security. The smart card figures very prominently in this. From the cardholder's standpoint, being able to have the smart card computer platform in physical possession is a large step toward overall security. In this case, attacks against the security of the overall system have to be made against the system components while in operation or through examination of information gained while the system was in operation. This means, for example, that attacking encryption algorithms used by the smart card must typically proceed from captured cipher text, not from active examination of the card while in use.

Conversely, the overall security architecture of the smart card-enabled system must be such that if a card is no longer in the cardholder's possession, the damage to the system through a security attack can be limited through the knowledge that the card is no longer in the cardholder's possession. Further, the vulnerability to the entire system must be minimized if the information related to a single cardholder is compromised.

3.5.2 Privacy

The final concept of security to be dealt with is privacy, which is keeping the details of a transaction secret from everyone not involved in the transaction. The cryptographic mechanisms previously discussed are adequate to provide transaction privacy. In general, the major design factor (that is, deciding which mechanism to actually use) is one of performance in the actual operational environment.

3.5.3 Access Control

Perhaps this is the starting point for any storage and processing system. The authorized user wishes to be assured that only the user can read and modify his personal data. In other words user wants controls to prevent unauthorized access to his data (or programs). In normal security terms control is subdivided into two terms, Logical access control, and Physical access control.

Logical access control concerns such familiar principles as password checking or the more sophisticated cryptographic mechanisms for authentication. Physical access control relates to the difficulty of a perpetrator physically breaking into the store of data. For example connecting wires to the disk drive directly and bypassing the rest of the computer completely.

3.5.4 Authentication and tamper resistance

The principle security service user are concerned with at this stage are authentication and tamper resistance. Authentication may relate to source authentication (i.e. confirmation of the identity of the source of communication) or peer entity authentication which assures one entity of the purported identity of the other correspondent. Tamper resistance is that fascinating feat of engineering that makes a device resistant to physical attack (proof against attack would be deemed an illusionary objective). As you might expect it is in this particularly difficult area that the IC chip offers significant benefits. It is probably true to say that it is also this objective that has been the most difficult to achieve adequately with conventional cryptographic equipment.

3.5.5 Data Integrity

For many applications and particularly in the financial world the preservation of data integrity is the principle security requirement. In this user are concerned with thwarting any event that results in the unauthorized tampering of the data. This includes not only modification of data but also addition or deletion of data.

3.5.6 Confidentiality

The need to preserve secrecy of data occurs in many applications. This relates both to the storage of data and the transmission of data. Where any cryptographic mechanisms are employed it is of course necessary to preserve the secret keys from compromise.

3.5.7 Non- Repudiation

Non - repudiation relates to that security service which ensures that a correspondent in some data interchange cannot subsequently deny his actions. Where trusted entities are communicating this facility is not required. It is clear however that in most cases this is not the practical case and there is invariably a need to be able to resolve disputes between the various parties involved in an interchange.

3.5.8 Audit Services

In the ideal environment the application of security mechanisms should be transparent to the users. This results in some difficulty in being assured that security controls are operating correctly. It is not too different to that situation where you jump out of an airplane with a parachute. It’s only then that you find out if it really works.

CHAPTER 4

SMART CARD READER ARCHITECTURE

4.1 Introduction

The Smart Card Reader/Writer is an interface for the communication between a computer (for example, a PC) and a smart card. Different types of smart cards have different commands and different communication protocols. This prevents in most cases the direct communication between a smart card and a computer. The Reader/Writer establishes a uniform interface from the computer to the smart card for a wide variety of cards. By taking care of the card specific particulars, it releases the computer software programmer of getting involved with the technical details of the smart card operation, which are in many cases not relevant for the implementation of a smart card system[13].

The Smart Card Reader/Writer is connected to the computer through a serial asynchronous interface (RS-232) or USB interface. The reader accepts commands from the computer, carries out the specified function at the smart card and returns the requested data or status information.

4.2 Features

ISO7816-1/2/3 compatible smart card interface, Supports CPU-based cards with T=0 and/or T=1 protocol. Supports commonly used memory cards (I2C, SLE4406, SLE4418/28, SLE4432/42), Support PPS (Protocol and Parameters Selection) with 9600 – 96000 bps in reading and writing smart cards, RS-232 interface or USB interface to PC with simple command structure.

4.3 Smart Card Interface

The interface between the Smart Card Reader/Writer and the inserted smart card uses the specifications of *ISO7816-3* with certain restrictions or enhancements to increase the practical functionality of the Smart Card Reader/Writer.

4.3.1 Smart Card Power Supply VCC (C1)

The current consumption of the inserted card must not be higher than 5**0mA**.

4.3.2 Programming Voltage VPP (C6)

According to ISO 7816-3, the smart card contact C6 (VPP) supplies the programming voltage to the smart card. The electrical specifications of this contact are identical to those of the signal RST (at contact C2).

4.3.3 Card Type Selection

The controlling PC has to always select the card type through the proper command sent to the Smart Card Reader/Writer prior to activating the inserted card. This includes both the memory cards and MCU-based cards.

For MCU-based cards the reader allows to select the preferred protocol, T=0 or T=1. However, this selection is only accepted and carried out by the reader through the PPS when the card inserted in the reader supports both protocol types. Whenever an MCU-based card supports only one protocol type, T=0 <u>or</u> T=1, the reader automatically uses that protocol type, regardless of the protocol type selected by the application.

4.3.4 Interface for Microcontroller-based Cards

For microcontroller-based smart cards only the contacts C1 (VCC), C2 (RST), C3 (CLK), C5 (GND) and C7 (I/O) are used. A frequency of 3.6864 / 4 MHz is applied to the CLK signal (C3).

4.3.5 Card Tearing Protection

The Smart Card Reader/Writer provides a mechanism to protect the inserted card when it is suddenly withdrawn while it is powered up. The power supply to the card and the signal lines between the Smart Card Reader/Writer and the card are immediately deactivated when the card is being removed. As a general rule, however, to avoid any electrical damage, a card should only be removed from the reader while it is powered down.

4.4 Power Supply

The Smart Card Reader/Writer requires a voltage of 5V DC, 100mA, regulated, power supply. The Smart Card Reader/Writer gets the power supply from PC (through the cable supplied along with each type of reader). Green LED on the front of the reader indicate the activation status of the smart card interface: Indicates power supply to the smart card is switched on, i.e., the smart card is activated.

4.5 Serial Interface

The Smart Card Reader/Writer is connected to a computer through a serial asynchronous interface using the RS-232 standard.

4.5.1 Communication Parameters

These communication parameters used by the Smart Card Reader/Writer and cannot be modified by the host computer:

Table 4:1 Communication Parameters

Transmission protocol	serial asynchronous
Parity	none
Data Bits	8
Stop Bits	1
Handshake	through CTS

The Smart Card Reader/Writer provides two means to select the transmission speed (baud rate) used by the reader in the normal operation, by hardware and/or by software.

4.5.2 Baud Rate Selection

The default hardware baud rate setting is 9600 bps. The *SET_PROTOCOL* command allows setting the transmission speed (baud rate) and a delay time inserted between the bytes transmitted by the reader to the PC. .Please note that the setting made with this command is volatile and will be lost when the reader is being reset or powered up next time.

4.5.3 Interface Wiring

For the communication between the Smart Card Reader/Writer and a computer, five lines of the RS-232 interface are used: RxD, TxD, CTS, DTR and GND.

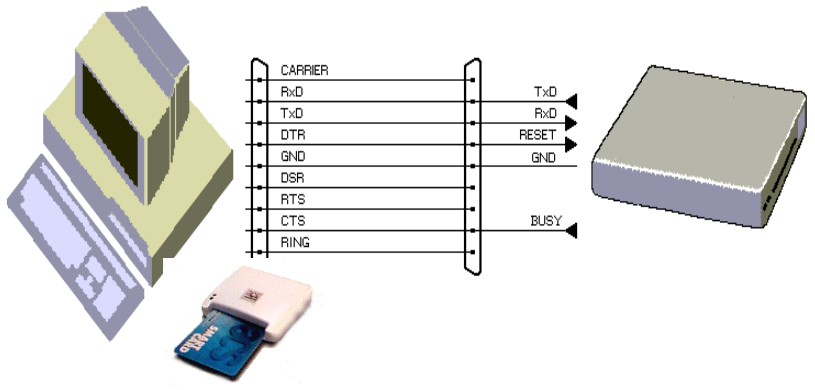


Figure 4:1 Interface Wiring

CHAPTER 5

SMART CARD OPERATING SYSTEM

5.1 Features

Features provided by the ACOS, 8 Kbytes of EEPROM memory for application data Compliance with ISO 7816-3, T=0 protocol, Five secret codes + Issuer Code PIN, changeable by card holder, Key pair for mutual authentication, Session key based on random numbers Linear files with fixed record length; record length can be different for different files[14].

5.2 Chip Life Cycle

During the whole life cycle of the chip-card, three phases and two different operating modes can be distinguished, Manufacturing Stage, Personalization Stage, User Stage, User Stage - Issuer Mode.

The card is at any moment in one of these four stages. the possible transitions between the four stages as shown in Figure 5:1

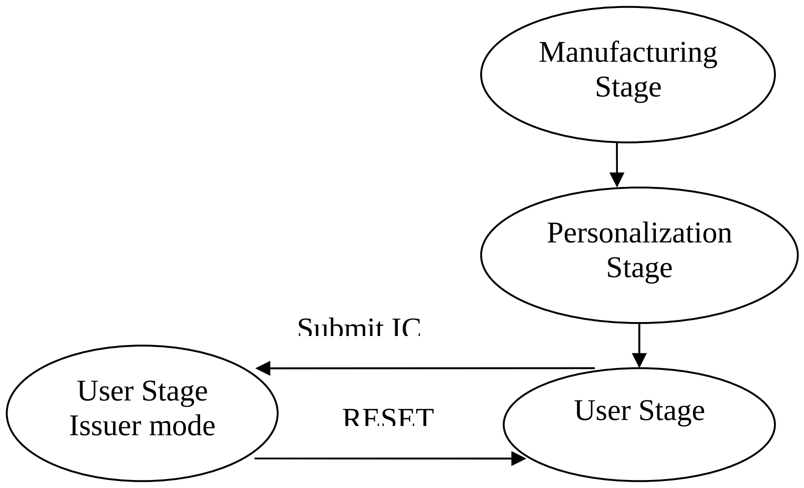


Figure 5:1 Chip Life Cycle

The actual chip life cycle stage is determined by the card operating system immediately after a reset. The life cycle stage does not change during the operation of the card.

5.2.1 Manufacturing Stage

The Manufacturing Stage is effective from the moment of chip manufacturing until an associated fuse (i.e., a certain bit in the EEPROM), the so-called *Manufacturer Fuse*, has been programmed.)

* * * **The blowing of the Manufacturer Fuse is irreversible** * * *

In the Manufacturing Stage, any write access to Internal Data Files, as well as the read access to the Security File is only possible after the presentation of the correct IC code. The initial IC code is programmed in the ACOS1 microcontroller during the chip manufacturing process.

The IC is presented to the card in plain, without encryption., All card commands are available, although some of the commands, such as AUTHENTICATE will not produce reasonable results as long as the respective data, for example, the keys, have not been programmed in the card.

The data items written to the EEPROM memory in the Manufacturing Stage are, The Manufacturer File, containing 2 records of 8 bytes each associated to the Manufacturing Stage. This file can only be written in the Manufacturing Stage. After programming the Manufacturer Fuse, the Manufacturer File is read-only. Data unique to each card and common card data can be programmed, such as, card manufacturer identification, card serial number, etc. The card does not interpret the data.

The IC code for the Personalization Stage. The IC code must have been presented to the card before the card allows WRITE access to the data files in the Personalization Stage, which is applicable immediately after completion of the Manufacturing Stage. The Manufacturer Fuse, to irreversibly change the card life cycle from the Manufacturing Stage to the Personalization Stage. The Manufacturer Fuse is one bit in the 16 bytes Manufacturer File.

5.2.2 Personalization Stage

The Personalization Stage is effective from the moment of termination of the Manufacturing Stage until an associated bit in the EEPROM, the so-called Personalization Bit, has been programmed. Once the Personalization Bit has been programmed and the Personalization Stage has thus been terminated, the Personalization Stage can be entered again from the Issuer Mode by executing the CLEAR CARD command. This command will physically erase the EEPROM memory, except for the Manufacturer Area, and thus return the card to the status it had before the Personalization Bit was initially programmed. Starting from ACOS1 revision 2.0, the CLEAR CARD command is disabled and so there is no way for a card in Issuer Stage to get back to the Personalization Stage.

In the Personalization Stage, any write access to Internal Data Files, as well as the read access to the Security File is only possible after the presentation of the correct IC code. The card manufacturer writes the IC code in the Manufacturing Stage.

The IC is presented to the card in plain, without encryption. The Authentication Process should not be executed prior to programming the correct keys in the Personalization Stage.

The data items written to the memory in the Personalization Stage are, The Personalization File, containing 3 records of 4 bytes each associated to the Personalization Stage, including the Option Registers. This area can only be written in the Personalization Stage. After programming the Personalization Fuse, the Personalization File is read-only. Data unique to each card and common card data can be programmed in the Personalization File, such as, card issuer identification, card application code, etc. The first 10 bytes of the Personalization File are transmitted in the Historical Bytes in the Answer-to-Reset.

Secret Codes and Keys. The File Definition Blocks of the required User Data Files. The Personalization Bit to change the card life cycle from the Personalization Stage to the User Stage.

5.2.3 User Stage

User Stage designates the 'normal' operating mode of the card. The User Stage is effective from the moment of termination of the Personalization Stage until the so-called Issuer Code has been submitted to the card. A submission of the Issuer Code changes the operation mode to the so-called Issuer Mode. This privileged mode allows access to certain memory areas, which are otherwise not accessible.

5.3 EEPROM Memory Management

The 8 k Bytes EEPROM memory area provided by the card chip is basically segregated in Internal Data Memory and User Data Memory: The Internal Data Memory is used for the storage of configuration data and it is used by the card operating system to manage certain functions. The User Data Memory stores the data manipulated in the normal use of the card under control of the application.

5.4 Data Files

Access to both the Internal Data Memory area and the User Data Memory area is possible within the scopes of data files and data records. Data files in the Internal Data Memory are referred to as Internal Data Files. Data files in the User Data Memory are called User Data Files.Data files are the smallest entity to which individual security attributes can be assigned to control the read and write access to the data stored in the EEPROM.

Data files are composed of data records. A data record is the smallest data unit that can individually be addressed in a data file. Each data file contains N data records. The record number must be specified when a record (or data within a record) is read from or written to a file. A data file can contain up to 255 records. The record length can be different for different files but is always fixed within a file. The file structures of the Internal Data Files (file size, file identifier, record length, security attributes) are defined by the operating system and cannot be changed. The file structure for the User Data Memory is determined in the card personalization. After programming the parameter N_OF_FILE in the Personalization Stage, the file structure is fixed.

Access to all files is possible only through the READ RECORD and WRITE RECORD commands. The operating system does not keep track of which records have actually been written through the WRITE RECORD command. The data returned by the card in response to a READ RECORD command are the actual data read from the EEPROM memory, regardless of whether that data have ever been written.

Each file is identified by two bytes File Identifier. The File Identifier is assigned to the file when the file is being defined during the Personalization Stage. The operating system does not perform any checking on the uniqueness of each File Identifier. If the same identifier has been assigned to more than one file, a malfunction of the card may occur.A value of FFH of the first byte of the file identifier is used for Internal Data Files and cannot be used for User Data Files.

Before any READ RECORD or WRITE RECORD access to a file, the file must be opened through the SELECT FILE command. Only one file is selected at any time. The READ RECORD and WRITE RECORD commands refer to the most recently selected file.

5.4.1 Data File Access Control

Two security attributes are assigned to each Data File: the Read Security Attribute and the Write Security Attribute. Security attributes define the security conditions that must be fulfilled to allow the respective operation, The Read Security Attribute controls the read access to the data in a file through the READ RECORD command. If the security condition specified in the Read Security Attribute is not fulfilled, the card will reject a READ RECORD command to that file.

The Write Security Attribute controls the write access to the data in a file through the WRITE RECORD command. If the security condition specified in the Write Security Attribute is not fulfilled, the card will reject a WRITE RECORD command to that file.

The Read Security Attribute and the Write Security Attribute for each data file specify which Application Code, if any, must have been submitted correctly to the card to allow the respective operation, and whether the Issuer Code and/or the PIN code must have been submitted.

A logical OR function applies to the specified Application Codes, AC x, i.e., if more than one Application Code is specified in a security attribute, the security condition is fulfilled if any one of the specified Application Codes has been correctly submitted. A logical AND function applies to the PIN and the IC code, i.e., if PIN and/or IC are specified in a security attribute, the PIN and/or IC code(s) must have been submitted in addition to the specified Application Codes(s).

Application Code AC0 can be specified in the Security Attribute, but cannot be submitted to the card. It is thus possible, for example, to completely write protect a file by specifying AC0 in the Write Security Attribute of that file.

For Internal Data Files, the security attributes are fixed in the card operating system. For User Data Files, the security attributes of a file are stored in the associated File Definition Block. The table 5.1 lists examples of security conditions that can be specified for User Data Files:

Table 5:1 Data File Access Control

Security Attribute	**Security Condition**
-	No restriction; free access
AC x	Access only after correct submission of AC x
AC x, AC y, AC z	Access only after correct submission of AC x or AC y or AC z
IC	Access only after submission of IC
PIN	Access only after submission of PIN
PIN, IC	Access only after submission of PIN and IC
AC x, IC	Access only after submission of AC x and IC
AC x, PIN, IC	Access only after submission of AC x, and PIN and IC
AC x, AC y, PIN	Access only after correct submission of AC x or AC y, and PIN
AC0	No access

AC x	requires Application Code x
PIN	requires PIN code
IC	requires Issuer Code

A Security Attribute is defined in one byte as shown Figure 5:2.

bit 7 bit 0

PIN	IC	AC5	AC4	AC3	AC2	AC1	AC0

Figure 5:2 Security Attribute

Each bit of the byte represents a code. If the bit is set to '1', the corresponding code must have been submitted. If the bit is set to'0', the corresponding code is irrelevant for the access condition.

5.4.2 MCU ID File

The MCU ID File contains two records of eight bytes each. The contents of this file are determined during the chip manufacturing process and cannot be altered.This file is always free for READ access but not WRITE accessible.

5.4.3 Manufacturer File

The Manufacturer File comprises two records of eight bytes each that are written in the Manufacturing Stage of the card life cycle. After termination of the Manufacturing Stage, this file is read-only and free for READ access.

The termination of the Manufacturer Stage is indicated by writing a '1' into the MSB of byte 1 of the first record in the Manufacturer File (Manufacturer Fuse). After the next reset of the card, the Manufacturing Stage can never again be entered.

5.4.4 Personalization File

The Personalization File comprises 12 bytes, arranged as 3 records of 4 bytes each. The Personalization File is written during the Personalization Stage of the card life cycle. After termination of the Personalization Stage, this file is read-only and free for READ access.

The termination of the Personalization Stage is indicated by writing a '1' into the MSB of byte 4 of the first record in the Personalization File (Personalization Bit). The change of stage will be effective immediately after the next reset of the card.

5.4.5 Security File

The information stored in the Security File, The key pair used for card authentication. The five Application Codes used for the file access control. The Issuer Code IC.

The PIN code., Error counters for limiting the number of unsuccessful code presentations and authentication. The Security File can only be read during the Manufacturing Stage and the Personalization Stage of the card life cycle, after presentation of the correct IC. After termination of the Personalization Stage, there is NO possibility to read the Security File.

5.4.6 User File Management File

The User File Management File consists of N_OF_FILE records of 6 bytes each and stores a File Definition Block for an allocated User Data File in each record. The File Definition Blocks are written during the Personalization Stage of the card life cycle. After termination of the Personalization Stage, this file is free for read access and can be written after the Issuer Code has been submitted.

The sequence of File Definition Blocks in the User File Management Area is not relevant. When the SELECT FILE command is issued, the card operating system searches all File Definition Blocks for one whose File Identifier entry matches the value specified in the SELECT FILE command.

The Card Operating System does not provide any error checking on the File Definition Blocks nor does it check the consistency of the number of file definition blocks written with the parameter N_OF_FILE. Any inconsistency of these data can lead to a malfunction of the card.

5.4.7 User File Data Area

The User File Data Area stores the data written to the User Data Files. Security attributes are attached to User Data Files, which control the access to the data in the files.User Data Files cannot be deleted. Once allocated, the memory space for a User Data File is reserved and cannot be released when the file is no longer used.

5.5 USER DATA FILES

User Data Files are allocated in the Personalization Stage of the card life cycle. The data stored in a User Data File can be read through the READ RECORD command and updated through the WRITE RECORD command when the security conditions associated to the data file are fulfilled.

User Data Files are defined by writing the corresponding File Definition Blocks in the records of the User File Management File during the Personalization Stage. It is not possible to change the number of records of a file once any of the User Data Files has been used. A User Data File can contain up to 255 records of max. 32 bytes record length each.

5.5.1 Memory space for User Data Files

The available memory space for User Data Files depends on whether or not the Account Data Structure is required. If the option bit ACCOUNT is set, the Account Data Structure reduces the memory space available for User Data Files by 64 bytes or 96 bytes, depending on Single DES or Triple DES is selected, respectively.

For maximum flexibility, the memory space occupied by the User File Management File is not fixed but depends on the number of User Data Files. Therefore, the amount of memory space available for User Data Files also depends on the number of files. Some memory space is required for each file for the internal management by the operating system. Available memory space for User Data Files is shown in Table 5:2.

Table 5:2 Memory space for User Data Files

SIZE	**Account Data Structure not Available (ACCOUNT = 0)**	**Account Data Structure Available with 1-DES (3-DES=0 and ACCOUNT =1)**	**Account Data Structure available with 3-DES (3-DES=1 and ACCOUNT = 1)**
No of files Even	7964 – N * 6	7900 – N * 6	7868 – N * 6
No of Files odd	7962 – N * 6	7898 – N * 6	7866 – N * 6

N = number of User Data Files defined

5.5.2 User File Definition Block

Each User Data File is described in an associated File Definition Block which contains the file identifier, record length, file length and security attributes. Each File Definition Block comprises six (6) bytes. Table 5:3 shows user file definition block.

Table 5:3 User File Definition Block

Byte 1	Byte 2	Byte 3	Byte 4	Byte 5 / 6
Record Length	Number of Records	Read Security Attribute	Write Security Attribute	File Identifier

The File Definition Blocks of all files are stored in the User File Management File. They can be read through READ RECORD commands after selection the User File Management File with the SELECT FILE command. The number of records in the User File Management File is given by the value of the parameter N_OF_FILE in the option register.

5.5.3 User File Allocation

For the allocation of User Data Files in a new card, use the steps as described. It is assumed that the IC has been presented to the card prior to this operation such that the Internal Data Files can be written.

Use the SELECT FILE command with file ID = FF02 to select the Personalization File. Write the number of User Data Files required to the option register N_OF_FILE, which is the third byte of the first record of the Personalization

File, to allocate the required space (number of records) in the User File Management File.

Use the SELECT FILE command with file ID = FF04 to select the User File Management File.Write the N_OF_FILE file definition blocks to the User File Management File with the WRITE RECORD command. Write the six bytes of each File Definition Block at once. Now the User Data Files can be selected and read and written.

5.6 DATA FILE ACCESS

The process of data file access is identical for Internal Data Files and for User Data Files.

5.6.1 SELECT FILE

The SELECT FILE command can be executed any time. The specified file - if existent - will be selected and the previously selected file - if any - will be closed. If the specified file does not exist, the card returns an error code and does not change the status of a currently selected file. The security conditions specified for the newly selected file are not checked in the SELECT FILE processing and the Mutual Authentication need not be completed prior to the execution of the SELECT FILE command. After a card reset, no file is selected.The SELECT FILE command is carried out as shown in Figure 5:2:

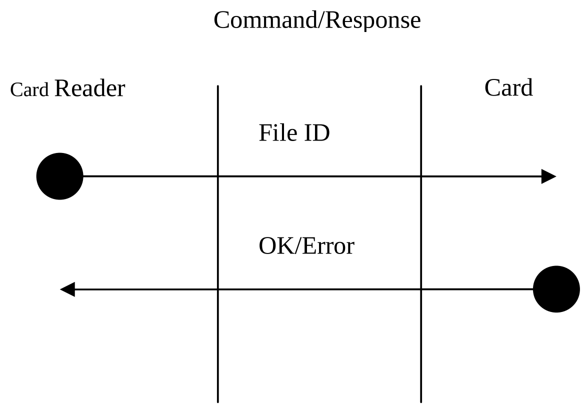


Figure 5:2 Select File

File ID	Two bytes file identifier of the file to be selected

5.6.2 READ RECORD

The READ RECORD command can be executed once a file has been selected through the SELECT FILE command. The security conditions associated to the currently selected file are checked prior to the execution of the command by the card. If the security conditions are not fulfilled (i.e., the specified secret codes have not been submitted to the card), the command is rejected by the card.

Data from only one record can be read in each READ RECORD operation. The number of bytes to be read is specified in the command. The maximum number of bytes to be read is equal to the record length. If the number of bytes read (= N) is smaller than the record length, the first N bytes of the record are returned by the card. The READ RECORD command is carried out as shown in Figure 5:3:

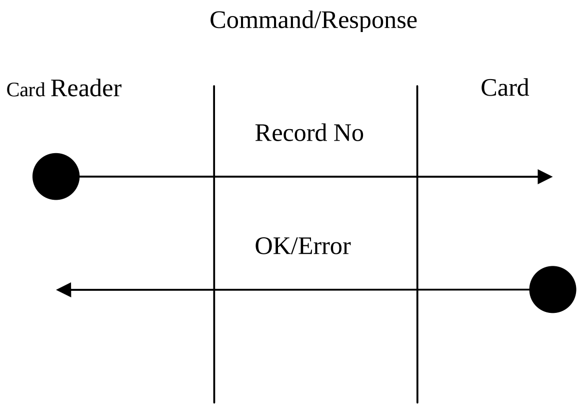


Figure 5:3 Read Record

Record No.	One byte logical record number
Length	Number of data bytes to be read from the record, max. 32
Data	Record data, *Length* bytes

5.6.3 WRITE RECORD

The WRITE RECORD command can be executed once a file has been selected through the SELECT FILE command. The security conditions associated to the currently selected file are checked prior to the execution of the command by the card. If the security conditions are not fulfilled (i.e., the specified secret codes have not been submitted to the card), the command is rejected by the card.

Data can be written to only one record in each WRITE RECORD operation. The number of bytes to be written in the record is specified in the command. The maximum number of bytes to be written is equal to the record length. If the number of bytes to be written (= N) is smaller than the record length, the first N bytes of the record are overwritten with the new data. The remaining bytes in the record are not modified. The WRITE RECORD command is carried out as shown in Figure 5:4.

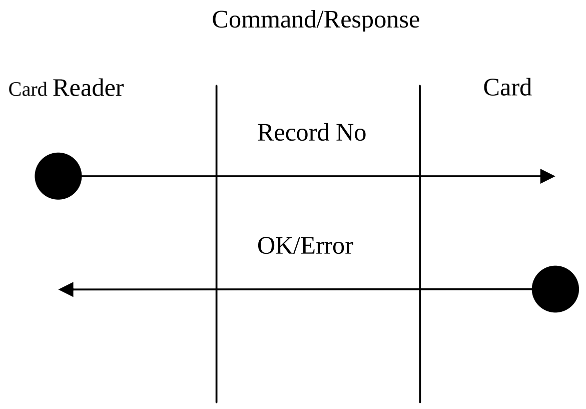


Figure 5:4 Write Record

Record No.	One byte logical record number
Data	Data bytes to be written to the record

5.7 SECRET CODES

Secret codes stored in the card are used to restrict the access to data stored in user data files and to certain commands provided by the card. Secret codes must be presented to the card in order to be able to read data from or write data to user data files and to execute certain privileged card commands. ACOS1 provides the secret codes, Five Application Codes (AC), One Issuer Code (IC), One PIN Code (PIN).

5.7.1 Application Codes

Five Application Codes (AC1 ... AC5) are available to control the access to the data stored in data files. Each Application Code is eight bytes long. An option bit in the Security Option Register in the Personalization File specifies for each code whether the code must be submitted to the card in plain or encrypted with the current session key.

5.7.2 Issuer Code

The Issuer Code is provided to control access to data files and to privileged card functions; it is eight bytes long., An option bit in the Security Option Register in the Personalization file specifies for the IC whether it must be submitted to the card in plain or encrypted with the current session key.

5.7.3 PIN code

The PIN Code is provided to control access to data files. The PIN is eight bytes long. The PIN is presented to the card with the SUBMIT CODE command. Depending on the corresponding option bit PIN_DES in the Security Option Register, the PIN is DES encrypted with the current session key before the presentation to the card, or it is presented in plain. The PIN code can be changed with the CHANGE PIN command if setting the PIN_ALT option bit in the option register has enabled this option. Depending on the option bit PIN_DES, the new code is DES encrypted with the current session key before it is written to the card, or it is written in plain.

5.7.4 Secret Code Submission and Error Counters

The card maintains an error counter CNT xx for each secret code to count and limit the number of consecutive unsuccessful executions of the SUBMIT CODE command, The error counter for a particular code is incremented by one each time the SUBMIT CODE operation for that code fails, i.e., a wrong secret code is submitted to the card., The error counter for a particular secret code is reset when the SUBMIT CODE operation for that code has successfully been executed. If the error counter reaches a value of eight (8), the card will reject the command SUBMIT CODE for that code.Depending on the setting of the corresponding bit in the Security Option Register, a code is submitted to the card as shown in Figure 5:5.

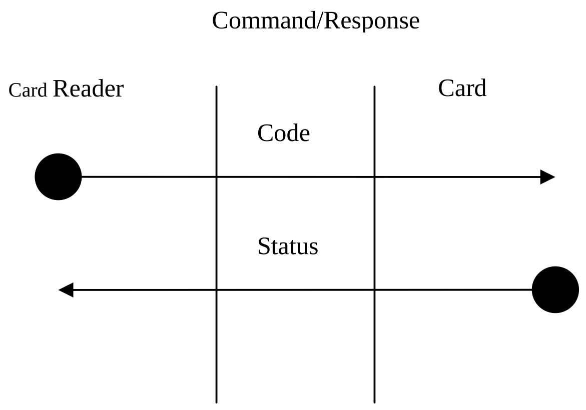


Figure 5:5 Secret Code Submission and Error Counters

Code Reference to the particular code that is submitted with the command:

1 ... 5	=	Application Codes AC1...AC5
6	=	PIN
7	=	Issuer Code IC

Other values for Code No. are not allowed and will be rejected by The Card Code. The eight bytes secret code to be submitted.

The error counters CNT xx are stored in the Security File. The counter value for a particular code is returned in the response by the card to an unsuccessful SUBMIT CODE operation. A backup copy of all error counters is kept in the Security File to prevent a corruption of this important information if an update in progress is interrupted through a reset of the card.

5.7.5 Change PIN code

The PIN code can be changed in the User stage with the command CHANGE PIN if the option bit PIN_ALT is set. To program a new PIN code in the card, the current PIN code must have been submitted first as shown in Figure 5:6.

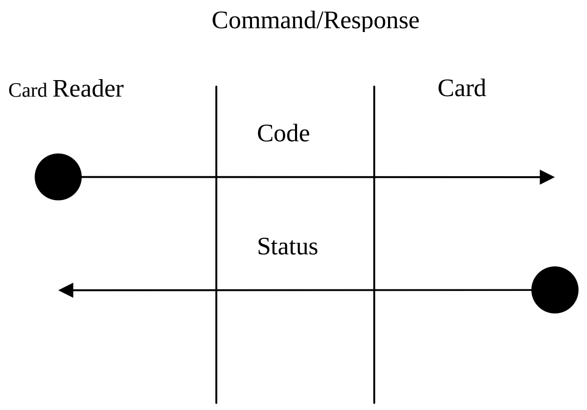


Figure 5:6 Change PIN code

5.8 Card Personalization

This section describes the general procedure in the personalization of a Smart Card. While the card personalization may be carried out in separate processing steps, the personalization process generally requires the execution of the steps in a sequence, Power up and reset the card, Submit the default Issuer Code IC (the code is communicated to the card issuer by ACS; the code may be different for different batches of cards supplied)

Select the Personalization File (File ID = FF 02H) and write the required settings to the Option Register and the parameter N_OF_FILE. Caution: Do not accidentally set the Personalization Bit and do not change the Security Option Register at this stage. Perform a card reset. After the reset, COS1 reads the Personalization File and accepts the new value of N_OF_FILE and the option bits stored in the Option Register.

Submit the default Issuer Code IC. Select the User File Management File (File ID = FF 04H) and write the File Definition Blocks for the required User Files (WRITE RECORD command) with the security attributes set to 'Free Access'.Select the individual User Files and initialize the data in the files as required (WRITE RECORD command).Select the User File Management File (File ID = FF 04H) and write the required security attributes for all User Files (WRITE RECORD command). Verify the contents of the User File Management File (READ RECORD command).

Caution: Do not accidentally change the other parameters in the File Definition Blocks.

If applicable, select the Account File (File ID = FF 05H) and initialize the relevant data in the Account File (WRITE RECORD command). Verify the contents of the Account File (READ RECORD command). If applicable, select the Account Security File (File ID = FF 06H) and initialize the account processing keys (WRITE RECORD command). Verify the contents of the Account Security File (READ RECORD command). Select the Security File (File ID = FF 03H) and initialize all keys and codes (WRITE RECORD command). Verify the contents of the Security File (READ RECORD command)

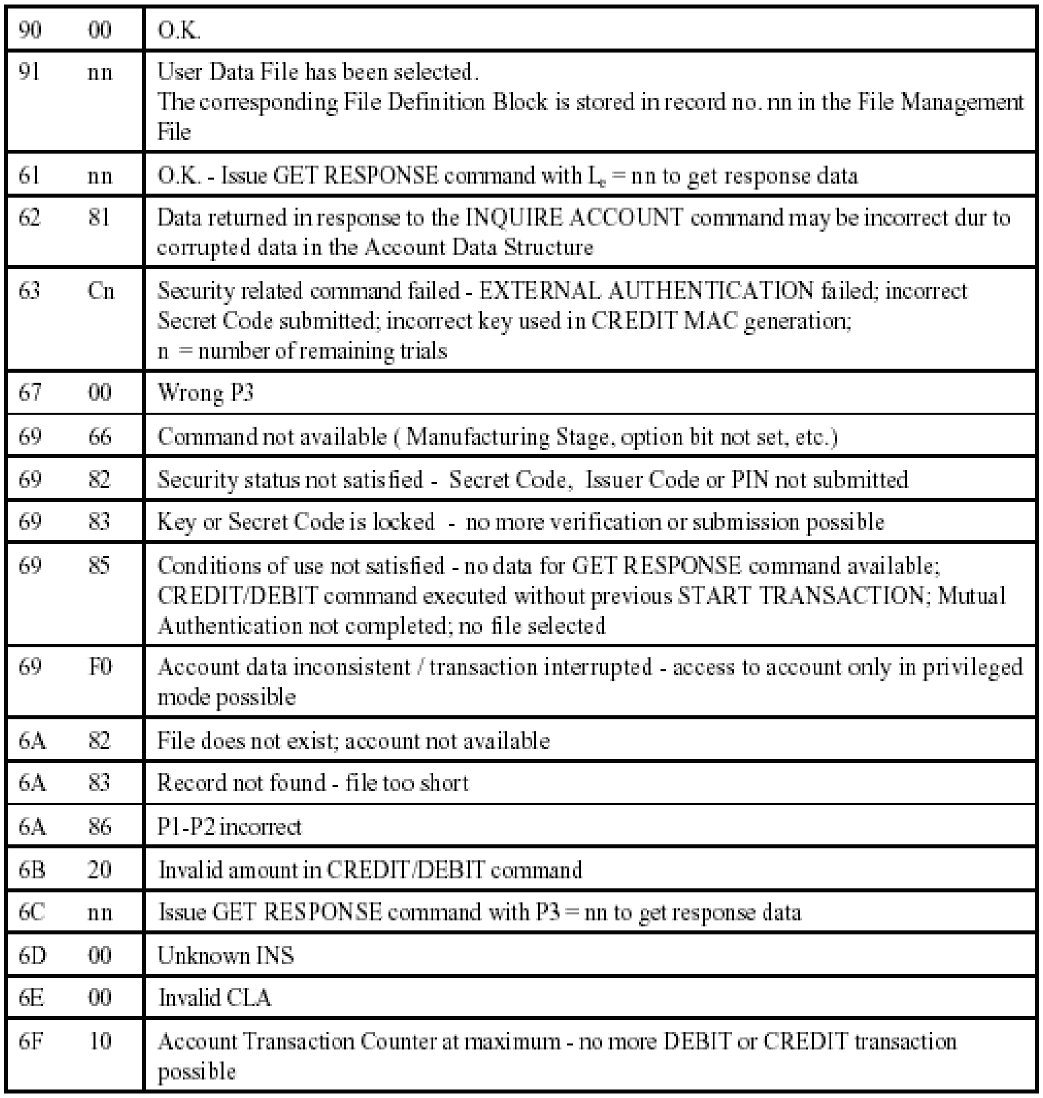
Select the Personalization File (File ID = FF 02H) and initialize the Security Option Register and the remaining bytes of the Personalization File. Set the Personalization Bit (WRITE RECORD command). Verify the contents of the Personalization File (READ RECORD command). Caution: Do not accidentally change the value of the Option Register and N_OF_FILE Figure 5:5.Perform a card reset. The chip life cycle stage as indicated in the ATR should be ‘User Stage’.

The correct personalization can be verified by submitting the secret codes and keys programmed in the card (SUBMIT CODE commands) and reading/writing the allocated data files and executing the Account commands.

5.9 Status Codes

The summary of the status codes returned by the card are shown in table5:4.

Table 5:4 Status Codes



90	00	O.K.
91	nn	User Data File has been selected. The corresponding File Definition Block is stored in record no. nn in the File Management File
61	nn	O.K. - Issue GET RESPONSE command with L_e = nn to get response data
62	81	Data returned in response to the INQUIRE ACCOUNT command may be incorrect dur to corrupted data in the Account Data Structure
63	Cn	Security related command failed - EXTERNAL AUTHENTICATION failed; incorrect Secret Code submitted; incorrect key used in CREDIT MAC generation; n = number of remaining trials
67	00	Wrong P3
69	66	Command not available (Manufacturing Stage, option bit not set, etc.)
69	82	Security status not satisfied - Secret Code, Issuer Code or PIN not submitted
69	83	Key or Secret Code is locked - no more verification or submission possible
69	85	Conditions of use not satisfied - no data for GET RESPONSE command available; CREDIT/DEBIT command executed without previous START TRANSACTION; Mutual Authentication not completed; no file selected
69	F0	Account data inconsistent / transaction interrupted - access to account only in privileged mode possible
6A	82	File does not exist; account not available
6A	83	Record not found - file too short
6A	86	P1-P2 incorrect
6B	20	Invalid amount in CREDIT/DEBIT command
6C	nn	Issue GET RESPONSE command with P3 = nn to get response data
6D	00	Unknown INS
6E	00	Invalid CLA
6F	10	Account Transaction Counter at maximum - no more DEBIT or CREDIT transaction possible

CHAPTER 6

SMART CARD INTERFACE SOFTWARE

6.1 Introduction Smart Card Interface Software

This chapter describes the use of smart card interface software to program the smart card readers. It is a set of library functions implemented for the application programmers to operate the smart card readers and the inserted smart cards. It can be programmed using the popular development tools like Visual C/C++, Borland C/C++, Visual Basic, Delphi, FoxPro, etc…

Depending on the reader model, smart card readers can be connected to the PC via the RS/232 interface or USB interface. The architecture of the Interface can be visualized as shown in Figure 6:1

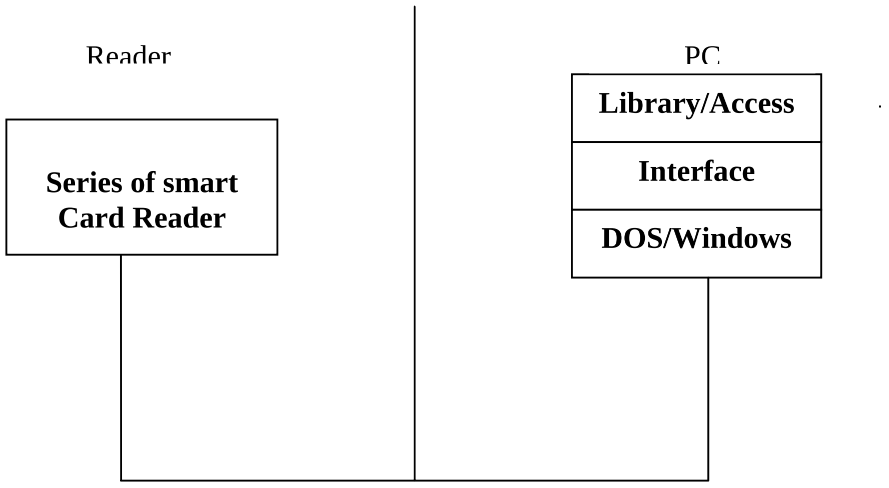


Figure 6:1 Architecture of the Interface

The interface controls the communication speed between the reader and the PC. For those readers using the serial RS232 connection, the default communication baud rate (factory setting) is 9600bps, no parity, eight bits and one stop bits. A higher speed of 115200bps can be achieved by using software command issuing from the host. If you are not sure about the factory setting of your readers, you can always use the ACR_AUTODETECT option in your AC_Open command for the driver to detect the communication speed automatically. Please notice that the communication speed setting apply only on those readers using the RS232 connection. For the PS/2 type and the USB type of connection, the speed is fixed at 9600bps and 1.5Mbps respectively.

6.2 Interface Data Structure

6.2.1 AC_APDU

```
typedef    struct {
    BYTE            CLA;
    BYTE            INS;
    BYTE            P1;
    BYTE            P2;
    INT16           Lc;
    INT16           Le;
    BYTE            DataIn[256];
    BYTE            DataOut[256];
    WORD16     Status;
} AC_APDU;
```

The AC_APDU data structure is used in the AC_ExchangeAPDU function for the passing of commands and data information into the smart card. Please notice that Lc representing the data length going into the card and Le representing the data length expecting from the card. The data types are defined in table 6:1.

Table 6:1 AC_APDU

Name	Input/Output	Description
CLA	I	Instruction Class
INS	I	Instruction Code
P1	I	Parameter 1
P2	I	Parameter 2
Lc	I	Length of command data (DataIn)
Le	I/O	Length of response data (DataOut)
DataIn	I	Command data buffer
DataOut	O	Response data buffer
Status	O	Execution status of the command

6.2.2 AC_SESSION

```
typedef    struct {
BYTE  CardType;       // Card type selected
 BYTE SCModule;     // Selected security module.
//Use only when card type = AC_SCModule
BYTE  ATRLen;        // Length of the ATR BYTE ATR[128];        // ATR string
BYTE  HistLen;         // Length of the Historical data
BYTE  HistOffset; // Offset of the Historical data
// from the beginning of ATR INT16 APDULenMax; // Max. APDU supported
} AC_SESSION;
```

The AC_SESSION data structure is used in the AC_StartSession function call for the retrieval of ATR information from the smart card. Before calling AC_StartSession, the program needs to specify the value of CardType. After calling the function, the ATR string can be found in ATR field and the length is stored in ATRLen.

Table 6:2 AC_SESSION

Name	Input/Output	Description
CardType	I	The card type selected for operation
SCModule	I	The security module selected for operation.
ATRLen	O	Length of the ATR string
ATR	O	Attention to reset (ATR) string
HistLen	O	Obsolete field – not used anymore
HistOffset	O	Obsolete field – not used anymore
APDULenMax	O	Obsolete field - not used anymore

6.2.3 AC_INFO

```
typedef    struct {
INT16 nMaxC;          // Maximum number of command data bytes
INT16 nMaxR;          // Maximum number of data bytes that
// can be requested in a response
INT16 CType;// The card types supported by the reader
BYTE  CStat;  // The status of the card reader
BYTE  CSel;   // The current selection of card type
BYTE  szRev[10];      // The 10 bytes firmware type and
// revision code
INT16  nLibVer;       // Library version
Long    lBaudRate; // Current Running Baud Rate
} AC_INFO;
```

The AC_INFO data structure is used in the AC_GetInfo function call for the retrieval of reader related information. Their meaning are described in Table 6:3:

Table 6:3 AC_INFO

Name	**Input/Output**	**Description**
nMaxC	0	The maximum number of command data byte (DataIn) that can be accepted in the ExchangeAPDU command.
nMaxR	0	The maximum number of response data byte (DataOut) that will be appeared in the ExchangeAPDU command
CType	0	The card types supported by the reader .

Cstat	0	The status of the card reader
		Bit0 = card present (1) or absent (0)
		Bit1 = card powered up (1) or powered down (0)

szRev[10]	0	The firmware revision code
nLibVer	0	Library version (e.g. 310 is equal to version 3.10)

CHAPTER 7

SQL SERVER (DATABASE ARCHITECTURE)

The database component of Microsoft® SQL Server™ 2000 is a Structured Query Language (SQL)–based, scalable, relational database.

7.1 SQL Server 2000 database component

7.1.1 Database

A database is similar to a data file in that it is a storage place for data. Like a data file, a database does not present information directly to a user; the user runs an application that accesses data from the database and presents it to the user in an understandable format.

Database systems are more powerful than data files in that data is more highly organized. In a well-designed database, there are no duplicate pieces of data that the user or application must update at the same time. Related pieces of data are grouped together in a single structure or record, and relationships can be defined between these structures and records[13].

When working with data files, an application must be coded to work with the specific structure of each data file. In contrast, a database contains a catalog that applications use to determine how data is organized. Generic database applications can use the catalog to present users with data from different databases dynamically, without being tied to a specific data format.

A database typically has two main parts: first, the files holding the physical database and second, the database management system (DBMS) software that applications use to access data. The DBMS is responsible for enforcing the database structure, including Maintaining relationships between data in the database. Ensuring that data is stored correctly, and that the rules defining data relationships are not violated. Recovering all data to a point of known consistency in case of system failures.

7.1.2 Relational Database

Although there are different ways to organize data in a database, relational databases are one of the most effective. Relational database systems are an application of mathematical set theory to the problem of effectively organizing data. In a relational database, data is collected into tables (called relations in relational theory).

A table represents some class of objects that are important to an organization. For example, a company may have a database with a table for employees, another table for customers, and another for stores. Each table is built of columns and rows (called attributes and tuples in relational theory). Each column represents some attribute of the object represented by the table. For example, an Employee table would typically have columns for attributes such as first name, last name, employee ID, department, pay grade, and job title. Each row represents an instance of the object represented by the table. For example, one row in the Employee table represents the employee who has employee ID 12345[16].

When organizing data into tables, you can usually find many different ways to define tables. Relational database theory defines a process called normalization, which ensures that the set of tables you define will organize your data effectively.

7.1.3 Scalable

SQL Server 2000 supports having a wide range of users access it at the same time. An instance of SQL Server 2000 includes the files that make up a set of databases and a copy of the DBMS software. Applications running on separate computers use a SQL Server 2000 communications component to transmit commands over a network to the SQL Server 2000 instance. When an application connects to an instance of SQL Server 2000, it can reference any of the databases in that instance that the user is authorized to access. The communication component also allows communication between an instance of SQL Server 2000 and an application running on the same computer. You can run multiple instances of SQL Server 2000 on a single computer.

SQL Server 2000 is designed to support the traffic of the largest Web sites or enterprise data processing systems. Instances of SQL Server 2000 running on large, multiprocessor servers are capable of supporting connections to thousands of users at the same time. The data in SQL Server tables can be partitioned across multiple servers, so that several multiprocessor computers can cooperate to support the database processing requirements of extremely large systems. These groups of database servers are called federations.

Although SQL Server 2000 is designed to work as the data storage engine for thousands of concurrent users who connect over a network, it is also capable of working as a stand-alone database directly on the same computer as an application. The scalability and ease-of-use features of SQL Server 2000 allow it to work efficiently on a single computer without consuming too many resources or requiring administrative work by the stand-alone user. The same features allow SQL Server 2000 to dynamically acquire the resources required to support thousands of users, while minimizing database administration and tuning. The SQL Server 2000 relational database engine dynamically tunes itself to acquire or free the appropriate computer resources required to support a varying load of users accessing an instance of SQL Server 2000 at any specific time. The SQL Server 2000 relational database engine has features to prevent the logical problems that occur if a user tries to read or modify data currently used by others.

7.1.4 Structured Query Language

To work with data in a database, you have to use a set of commands and statements (language) defined by the DBMS software. Several different languages can be used with relational databases; the most common is SQL. The American National Standards Institute (ANSI) and the International Standards Organization (ISO) define software standards, including standards for the SQL language. SQL Server 2000 supports the Entry Level of SQL-92, the SQL standard published by ANSI and ISO in 1992. The dialect of SQL supported by Microsoft SQL Server is called Transact-SQL (T-SQL). T-SQL is the primary language used by Microsoft SQL Server applications.

7.2 Database Applications and Servers

Microsoft SQL Server 2000 is designed to work effectively as a central database on a server shared by many users who connect to it over a network. The number of users can range from a handful in one workgroup, to thousands of employees in a large enterprise, to hundreds of thousands of Web users. A desktop database that services only applications running on the same desktop.

7.2.1 Server Database Systems

Server-based systems are constructed so that a database on a central computer, known as a server, is shared among multiple users. Users access the server through an application, In a multitier system, such as Windows® DNA, the client application logic is run in two or more locations: A thin client is run on the user's local computer and is focused on displaying results to the user. The business logic is located in server applications running on a server. Thin clients request functions from the server application, which is itself a multithreaded application capable of working with many concurrent users. The server application is the one that opens connections to the database server. The server application can be running on the same server as the database, or it can connect across the network to a separate server operating as a database server. In complex systems, the business logic may be implemented in several interconnected server applications, or in multiple layers of server applications.

This is a typical scenario for an Internet application. For example, a multithreaded server application can run on a Microsoft® Internet Information Services (IIS) server and service thousands of thin clients running on the Internet or an intranet. The server application uses a pool of connections to communicate with one or more instances of SQL Server 2000. The instances of SQL Server 2000 can be on the same computer as IIS, or they can be on separate servers in the network.

In a two-tier client/server system, users run an application on their local computer, known as a client application, that connects over a network to an instance of SQL Server 2000 running on a server computer. The client application runs both business logic and the code to display output to the user, so this is sometimes referred to as a thick client.

7.2.2 Advantages of Server Database System

Having data stored and managed in a central location offers several advantages, Each data item is stored in a central location where all users can work with it. Separate copies of the item are not stored on each client, which eliminates problems with users having to ensure they are all working with the same information. Their system does not need to ensure that all copies of the data are updated with the current values, because there is only one copy in the central location. Business and security rules can be defined one time on the server and enforced equally among all users.

Rule enforcement can be done in a database through the use of constraints, stored procedures, and triggers. Rules can also be enforced in a server application, since these applications are also central resources accessed by many thin clients. A relational database server optimizes network traffic by returning only the data an application needs.

For example, if an application working with a file server needs to display a list of the names of sales representatives in Oregon, it must retrieve the entire employee file. If the application is working with a relational database server, it sends this command:

```
SELECT first_name, last_name
FROM employees
WHERE emp_title = 'Sales Representative'
 AND emp_state = 'OR'
```

The relational database sends back only the names of the sales representatives in Oregon, not all of the information about all employees. Hardware costs can be minimized. Because the data is not stored on each client, clients do not have to dedicate disk space to storing data. The clients also do not need the processing capacity to manage data locally, and the server does not need to dedicate processing power to displaying data.The server can be configured to optimize the disk I/O capacities needed to retrieve data, and clients can be configured to optimize the formatting and display of data retrieved from the server.

The server can be stored in a relatively secure location and equipped with devices such as an Uninterruptible Power Supply more economically than fully protecting each client. Maintenance tasks such as backing up and restoring data are simplified because they can focus on the central server.

7.2.3 Advantages of SQL Server 2000 as a Database Server

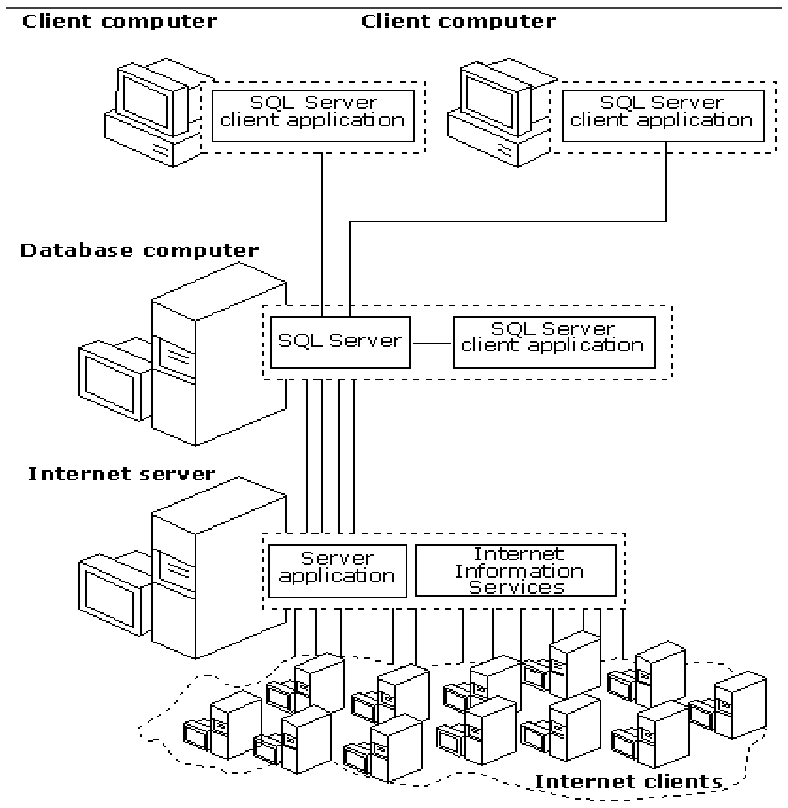
Microsoft SQL Server 2000 is capable of supplying the database services needed by extremely large systems. Large servers may have thousands of users connected to an instance of SQL Server 2000 at the same time. SQL Server 2000 has full protection for these environments, with safeguards that prevent problems, such as having multiple users trying to update the same piece of data at the same time. SQL Server 2000 also allocates the available resources effectively, such as memory, network bandwidth, and disk I/O, among the multiple users.

Extremely large Internet sites can partition their data across multiple servers, spreading the processing load across many computers, and allowing the site to serve thousands of concurrent users.

Multiple instances of SQL Server 2000 can be run on a single computer. For example, an organization that provides database services to many other organizations can run a separate instance of SQL Server 2000 for each customer organization, all on one computer. This isolates the data for each customer organization, while allowing the service organization to reduce costs by only having to administer one server computer.

SQL Server 2000 applications can run on the same computer as SQL Server 2000. The application connects to SQL Server 2000 using Windows Interprocess Communications (IPC) components, such as shared memory, instead of a network. This allows SQL Server 2000 to be used on small systems where an application must store its data locally.

The illustration shows an instance of SQL Server 2000 operating as the database server for both a large Web site and a legacy client/server system as shown in Figure 7:1.


Client computer
Client computer
SQL Server client application
SQL Server client application
Database computer
SQL Server
SQL Server client application
Internet server
Server application
Internet Information Services
Internet clients

.

Figure 7:1 Database Server

7.3 SQL Server Authentication Modes

When SQL Server 2000 is running on Windows NT or Windows 2000, members of the sysadmin fixed server role can specify one of two authentication modes:

7.3.1 Windows Authentication Mode

Only Windows Authentication is allowed. Users cannot specify a SQL Server 2000 login ID. This is the default authentication mode for SQL Server 2000. You cannot specify Windows Authentication Mode for an instance of SQL Server running on Windows 98, because the operating system does not support Windows Authentication.

7.3.2 Mixed Mode

If users supply a SQL Server 2000 login ID when they log on, they are authenticated using SQL Server Authentication. If they do not supply a SQL Server 2000 login ID, or request Windows Authentication, they are authenticated using Windows Authentication

7.4 Data Types

Because each column represents one attribute of an object, the data in each occurrence of the column is similar. One of the properties of a column is called its data type, which defines the type of data the column can hold. SQL Server has several basic data types that can be specified for columnsare shown on table7:1.

Table 7:1 Data types

binary	Bigint	bit	Char	datetime
decimal	Float	image	Int	Money
nchar	Ntext	nvarchar	Numeric	Real
smalldatetime	smallint	smallmoney	sql_variant	sysname
text	timestamp	tinyint	varbinary	varchar

7.5 System Tables

SQL Server stores the data defining the configuration of the server and all its tables in a special set of tables known as system tables. Users should not query or update the system tables directly unless there is no other way to get the data required by the application. Only SQL Server should reference the system tables in response to administration commands issued by users. The system tables can change from version to version; applications referencing system tables directly may have to be rewritten before they can be upgraded to a newer version of SQL Server with a different version of the system tables. SQL Server exposes most of the information from the system tables through other means.

7.6 SQL Stored Procedures

A stored procedure is a group of Transact-SQL statements compiled into a single execution plan. Microsoft® SQL Server™ 2000 stored procedures return data in four ways: Output parameters, which can return either data (such as an integer or character value) or a cursor variable (cursors are result sets that can be retrieved one row at a time).Return codes, which are always an integer value.A result set for each SELECT statement contained in the stored procedure or any other stored procedures called by the stored procedure.

A global cursor that can be referenced outside the stored procedure. Stored procedures assist in achieving a consistent implementation of logic across applications. The SQL statements and logic needed to perform a commonly performed task can be designed, coded, and tested once in a stored procedure. Each application needing to perform that task can then simply execute the stored procedure. Coding business logic into a single stored procedure also offers a single point of control for ensuring that business rules are correctly enforced. Stored procedures can also improve performance. Many tasks are implemented as a series of SQL statements.

7.7 ER-Diagram

7.7.1 Relationships in a Database Diagram

Within a database diagram, each relationship can appear with three distinct features: endpoints, a line style, and related tables.

7.7.1.1 Endpoints

The endpoints of the line indicate whether the relationship is one-to-one or one-to-many. If a relationship has a key at one endpoint and a figure-eight at the other, it is a one-to-many relationship. If a relationship has a key at each endpoint, it is a one-to-one relationship.

7.7.1.2 Line Style

The line itself (not its endpoints) indicates whether the Database Management System (DBMS) enforces referential integrity for the relationship when new data is added to the foreign-key table. If the line appears solid, the DBMS enforces referential integrity for the relationship when rows are added or modified in the foreign-key table. If the line appears dotted, the DBMS does not enforce referential integrity for the relationship when rows are added or modified in the foreign-key table.

7.7.1.3 Related Tables

The relationship line indicates that a foreign-key relationship exists between one table and another. For a one-to-many relationship, the foreign-key table is the table near the line's figure-eight symbol. If both endpoints of the line attach to the same table, the relationship is a reflexive relationship.

CHAPTER 8

DATABASE INTEGRATION WITH SMART CARDS

8.1 Accessing a Database

Relational databases are accessed by using some sort of database scripting language. The most commonly used database language is the Structured Query Language (SQL), which is used to manage not only databases on desktop computers but also huge databases used by banks, schools, corporations, and other institutions with sophisticated database needs. By using a language such as SQL, you can compare information in the various tables of a relational database and extract results made up of data fields from one or more tables combined.

8.2 The Visual C++ ODBC Classes

When you create a database program with Visual C++'s AppWizard, you end up with an application that draws extensively on the various ODBC classes that have been incorporated into MFC. The most important of these classes are CDatabase, CRecordset, and CRecordView.

AppWizard automatically generates the code needed to create an object of the CDatabase class. This object represents the connection between your application and the data source that you will be accessing. In most cases, using the CDatabase class in an AppWizard-generated program is transparent to you, the programmer. All the details are handled by the framework.

AppWizard also generates the code needed to create a CRecordset object for the application. The CRecordset object represents the actual data currently selected from the data source, and its member functions manipulate the data from the database.

Finally, the CRecordView object in your database program takes the place of the normal view window you're accustomed to using in AppWizard-generated applications. A CRecordView window is like a dialog box that's being used as the application's display. This dialog box-type of window retains a connection to the application's CRecordset object, hustling data back and forth between the program, the window's controls, and the recordset. When you first create a new database application with AppWizard, it's up to you to add edit controls to the CRecordView window. These edit controls must be bound to the database fields they represent so that the application framework knows where to display the data you want to view.

8.3 Creating an ODBC Database Program

Although creating a simple ODBC database program is easy with Visual C++, there are a number of steps you must complete, Register the database with the system. Use AppWizard to create the basic database application.Add code to the basic application to implement features not automatically supported by AppWizard**.**

8.3.1 Registering the Database

Before you can create a database application, you must register the database that you want to access as a data source that you can access through the ODBC driver. Use these steps to accomplish this important task: From the Windows Start menu, click Settings and then Control Panel. When the Control Panel dialog appears, double-click the 32-Bit ODBC icon. The ODBC Data Source Administrator dialog box appears, as shown in Figure 8:1.

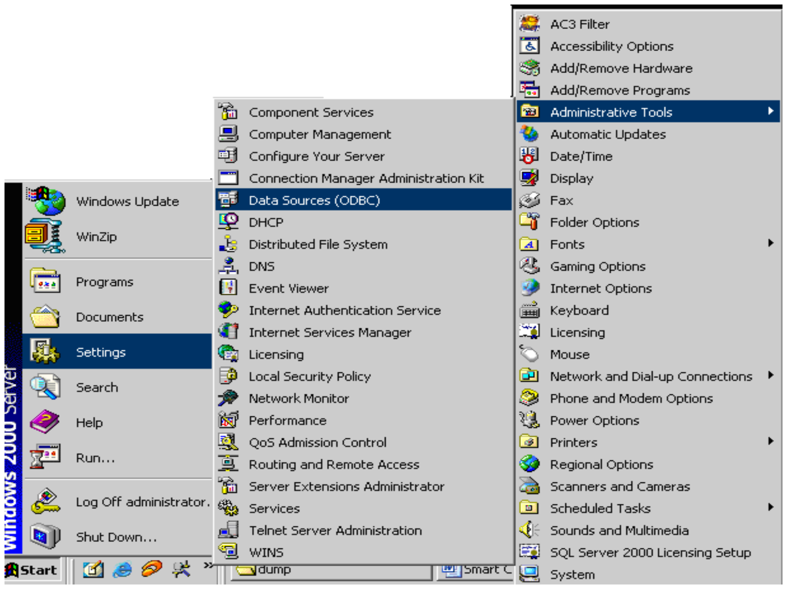
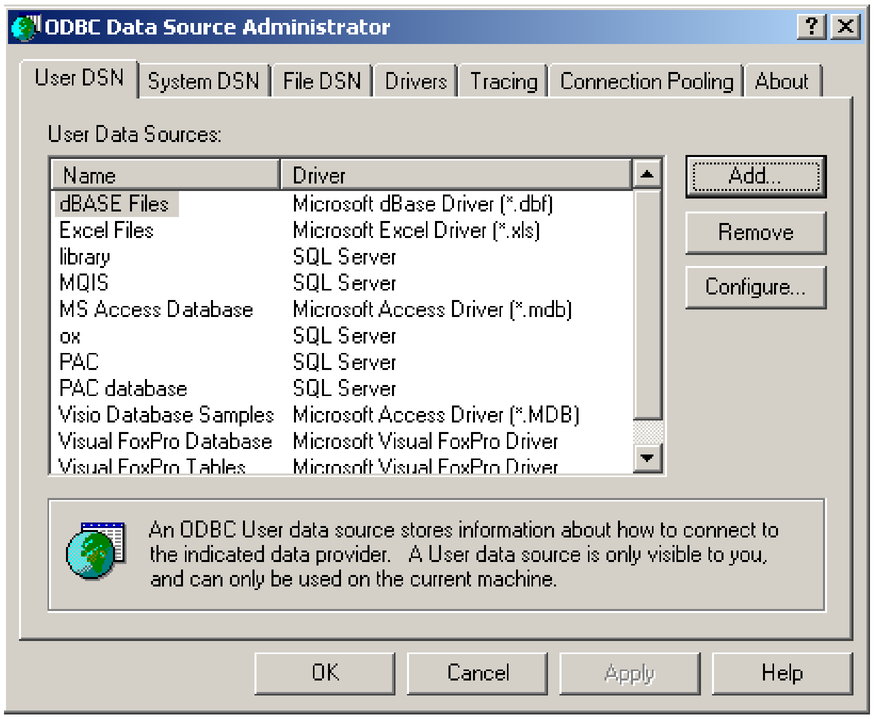


Figure 8:1 ODBC

Click the Add button. The Create New Data Source dialog box appears. Select the **SQL Server** Driver from the list of drivers, as shown in Figure 8:2 & Figure 8:3, and click Finish.


ODBC Data Source Administrator

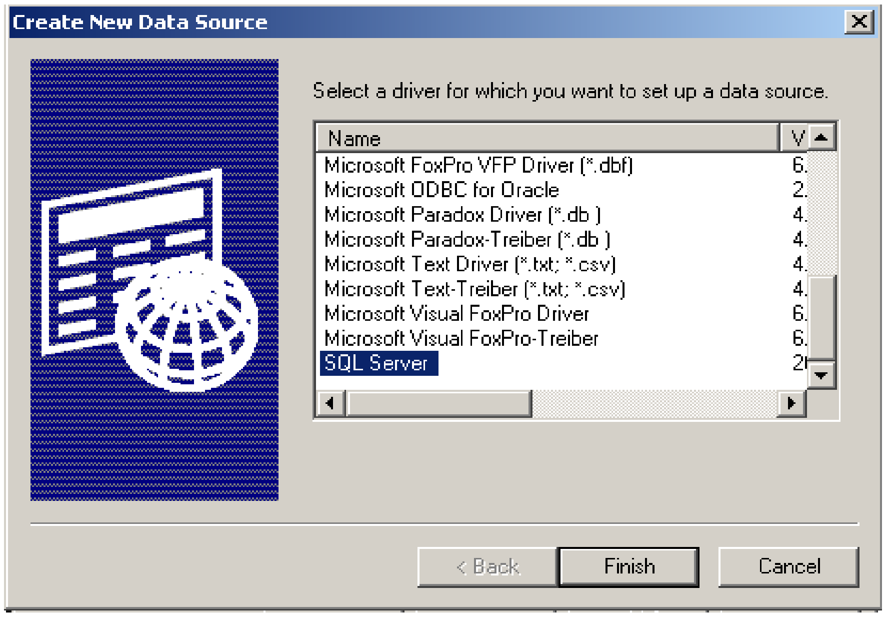
User DSN | System DSN | File DSN | Drivers | Tracing | Connection Pooling | About

User Data Sources:

Name	Driver
dBASE Files	Microsoft dBase Driver (*.dbf)
Excel Files	Microsoft Excel Driver (*.xls)
library	SQL Server
MQIS	SQL Server
MS Access Database	Microsoft Access Driver (*.mdb)
ox	SQL Server
PAC	SQL Server
PAC database	SQL Server
Visio Database Samples	Microsoft Access Driver (*.MDB)
Visual FoxPro Database	Microsoft Visual FoxPro Driver
Visual FoxPro Tables	Microsoft Visual FoxPro Driver

An ODBC User data source stores information about how to connect to the indicated data provider. A User data source is only visible to you, and can only be used on the current machine.

Figure 8:2 ODBC (ADD)

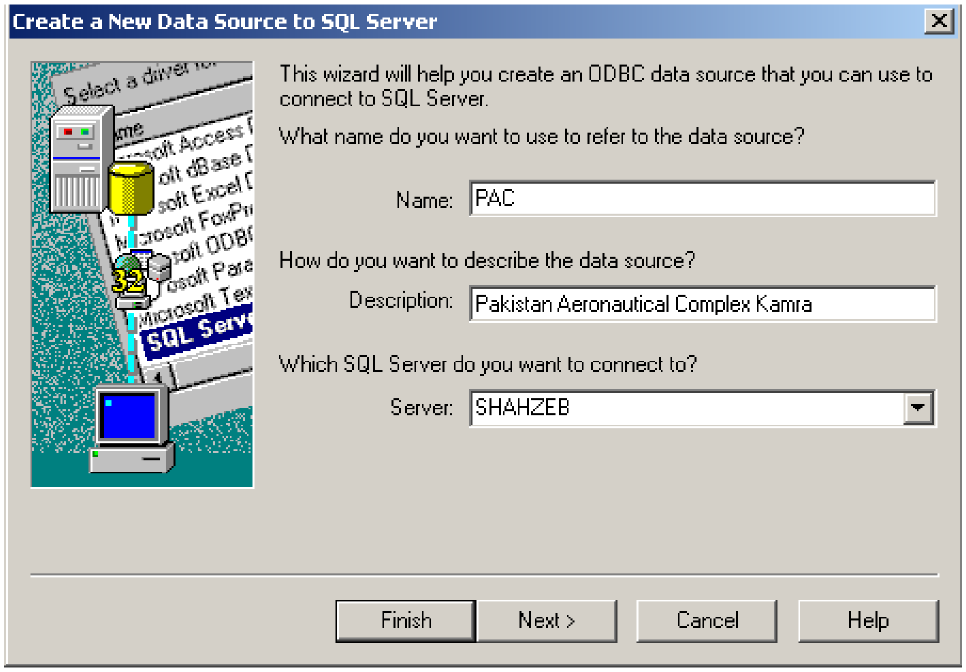

Create New Data Source

Select a driver for which you want to set up a data source.

Name	V
Microsoft FoxPro VFP Driver (*.dbf)	6.
Microsoft ODBC for Oracle	2.
Microsoft Paradox Driver (*.db)	4.
Microsoft Paradox-Treiber (*.db)	4.
Microsoft Text Driver (*.txt; *.csv)	4.
Microsoft Text-Treiber (*.txt; *.csv)	4.
Microsoft Visual FoxPro Driver	6.
Microsoft Visual FoxPro-Treiber	6.
SQL Server	2

Figure 8:3 SQL Server:

The SQL Server Driver is now the ODBC driver that will be associated with the data source you create for the Employee application.When the ODBC Microsoft SQL Server Setup dialog box appears, enter **PAC** in the Data Source Name text box and **Pakistan Aeronautical Complex Kamra** in the Description text box, as shown in Figure 8:4. The Data Source Name is a way of identifying the specific data source you're creating. The Description field enables you to include more specific information about the data source.


Create a New Data Source to SQL Server

This wizard will help you create an ODBC data source that you can use to connect to SQL Server.

What name do you want to use to refer to the data source?

Name: PAC

How do you want to describe the data source?

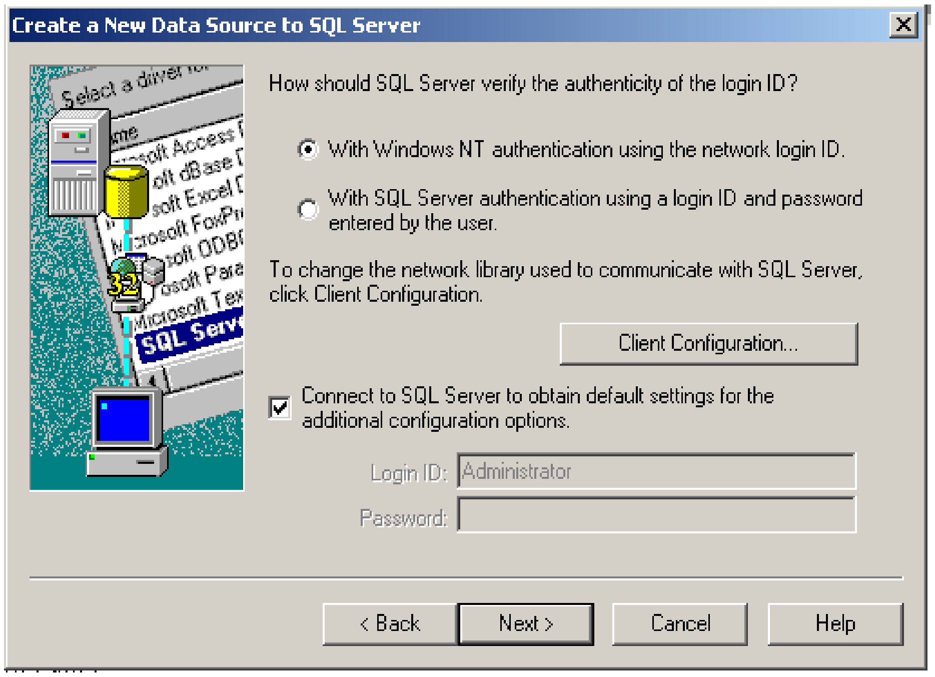
Description: Pakistan Aeronautical Complex Kamra

Which SQL Server do you want to connect to?

Server: SHAHZEB

Figure 8:4 PAC

Click the Next button. The Authentication box will appear select the windows NT authentication using Network login ID mode. As shown in Figure 8:5.



How should SQL Server verify the authenticity of the login ID?

(●) With Windows NT authentication using the network login ID.

( ) With SQL Server authentication using a login ID and password entered by the user.

To change the network library used to communicate with SQL Server, click Client Configuration.

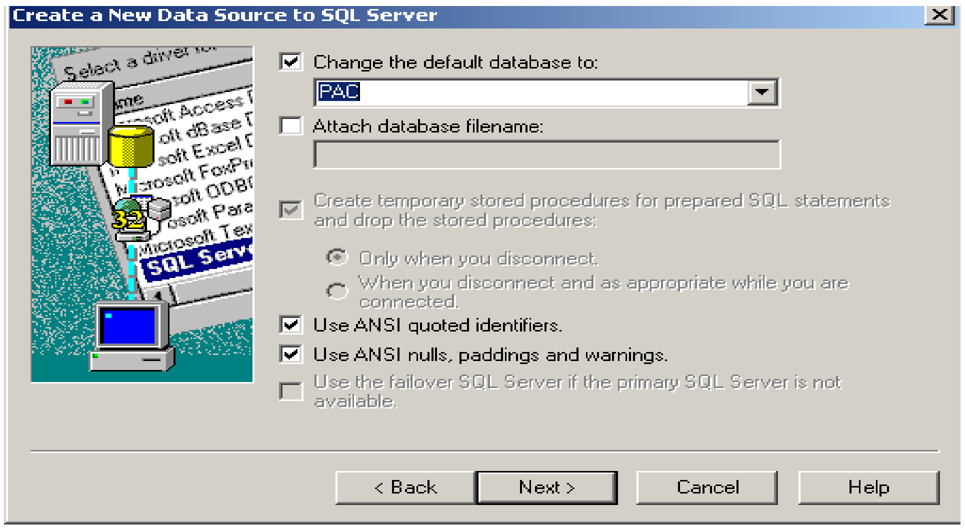
[✓] Connect to SQL Server to obtain default settings for the additional configuration options.

Login ID: Administrator

Password:

Figure 8:5 Authentication

Click the Next button. The Select Database file selector appears. Use the selector to select the PAC file as shown in Figure 8:6.



[✓] Change the default database to: PAC

[ ] Attach database filename:

[✓] Create temporary stored procedures for prepared SQL statements and drop the stored procedures:

(●) Only when you disconnect.

( ) When you disconnect and as appropriate while you are connected.

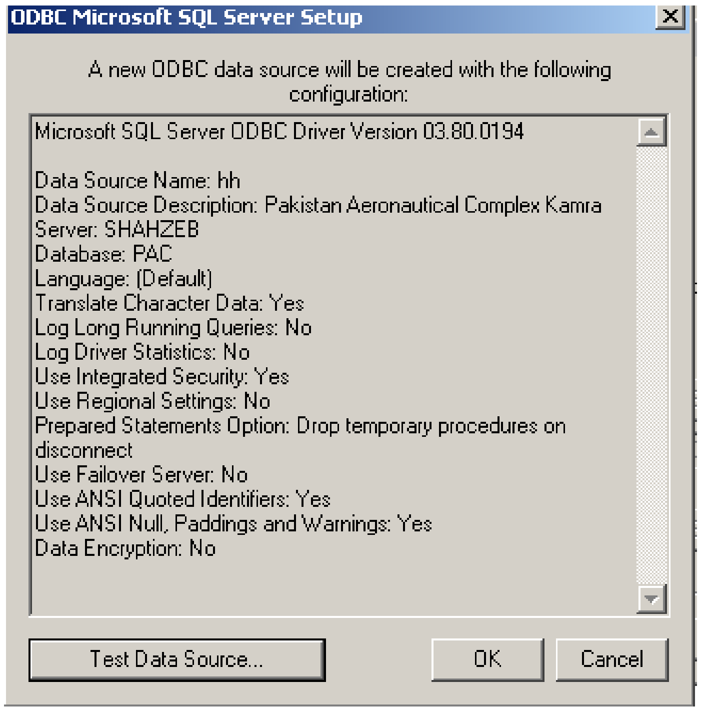
[✓] Use ANSI quoted identifiers.

[✓] Use ANSI nulls, paddings and warnings.

[ ] Use the failover SQL Server if the primary SQL Server is not available.

Figure 8:6 File Select

Click Next and then Finish to finalize the database selection and then, in the ODBC SQL server Setup dialog box, test data source dialog box will appear(See Figure 8:7). Test the data source. Click OK to finalize the data-source creation process. Finally, click OK in the ODBC Data Source Administrator dialog box.


ODBC Microsoft SQL Server Setup

A new ODBC data source will be created with the following configuration:

```
Microsoft SQL Server ODBC Driver Version 03.80.0194

Data Source Name: hh
Data Source Description: Pakistan Aeronautical Complex Kamra
Server: SHAHZEB
Database: PAC
Language: (Default)
Translate Character Data: Yes
Log Long Running Queries: No
Log Driver Statistics: No
Use Integrated Security: Yes
Use Regional Settings: No
Prepared Statements Option: Drop temporary procedures on
disconnect
Use Failover Server: No
Use ANSI Quoted Identifiers: Yes
Use ANSI Null, Paddings and Warnings: Yes
Data Encryption: No
```

Figure 8:7 Test Data source

8.4 Adding and Deleting Records

Adding and deleting records from a database table through MFC Visual C++ GUI (Graphical User Interface). The functions used by database are described in this section.

8.4.1 OnRecordAdd() Function

```
void CPACView::OnRecordAdd()
{
   m_pSet->AddNew();
   m_bAdding = TRUE;
   CEdit* pCtrl = (CEdit*)GetDlgItem(IDC_EMPLOYEE_ID);
   int result = pCtrl->SetReadOnly(FALSE);
   UpdateData(FALSE);
}
```

OnRecordAdd() starts with a call to the AddNew() member function of CEmployeeSet, the class derived from CRecordSet. This sets up a blank record for the user to fill in, but the new blank record doesn't appear on the screen until the view window's UpdateData() function is called. Before that happens, you have a few other things to tackle.

After the user has created a new record, the database will need to be updated. By setting a flag in this routine, the move routine will be able to determine whether the user is moving away from an ordinary database record or a newly added one. That's why m_bAdding is set to TRUE in function.

Now, because the user is entering a new record, it should be possible to change the contents of the Employee ID field, which is currently set to read-only. To change the read-only status of the control, the program first obtains a pointer to the control with GetDlgItem() and then calls the control's SetReadOnly() member function to set the read-only attribute to FALSE. Finally, the call to UpdateData() will display the new blank record.

8.4.2 OnMove()

```
BOOL CPACView::OnMove(UINT nIDMoveCommand)
{
    if (m_bAdding)
    {
        m_bAdding = FALSE;
        UpdateData(TRUE);
        if (m_pSet->CanUpdate())
            m_pSet->Update();
        m_pSet->Requery();
        UpdateData(FALSE);
        CEdit* pCtrl = (CEdit*)GetDlgItem(IDC_EMPLOYEE_ID);
        pCtrl->SetReadOnly(TRUE);
        return TRUE;
    }
    else
        return CRecordView::OnMove(nIDMoveCommand);
}
```

Now that the user has a blank record on the screen, it's a simple matter to fill in the edit controls with the necessary data. To add the new record to the database, the user must move to a new record, an action that forces a call to the view window's OnMove() member function. Normally, OnMove() does nothing more than display the next record. Your override will save new records as well.

When OnMove() is called, the first thing the program does is check the Boolean variable m_bAdding to see whether the user is in the process of adding a new record. If m_bAdding is FALSE, the body of the if statement is skipped and the else clause is executed. In the else clause, the program calls the base class (CRecordView) version of OnMove(), which simply moves to the next record.

If m_bAdding is TRUE, the body of the if statement is executed. There, the program first resets the m_bAdding flag and then calls UpdateData() to transfer data out of the view window's controls and into the recordset class. A call to the recordset's CanUpdate() method determines whether it's okay to update the data source, after which a call to the recordset's Update() member function adds the new record to the data source.

To rebuild the recordset, the program must call the recordset's Requery() member function, and then a call to the view window's UpdateData() member function transfers new data to the window's controls. Finally, the program sets the Employee ID field back to read-only, with another call to GetDlgItem() and SetReadOnly().

8.4.3 OnRecordDelete()

```
void CPACView::OnRecordDelete()
{       m_pSet->Delete();
        m_pSet->MoveNext();
        if (m_pSet->IsEOF())
              m_pSet->MoveLast();
        if (m_pSet->IsBOF())
              m_pSet->SetFieldNull(NULL);
        UpdateData(FALSE);
}
```

Deleting a record is simple. OnRecordDelete() just calls the recordset's Delete() function. When the record is deleted, a call to the recordset's MoveNext() arranges for the record that is next to be displayed. A problem might arise, though, when the deleted record was in the last position or when the deleted record was the only record in the recordset. A call to the recordset's IsEOF() function will determine whether the recordset was at the end. If the call to IsEOF() returns TRUE, the recordset needs to be repositioned on the last record. The recordset's MoveLast() function takes care of this task.

When all records have been deleted from the recordset, the record pointer will be at the beginning of the set. The program can test for this situation by calling the recordset's IsBOF() function. If this function returns TRUE, the program sets the current record's fields to NULL. Finally, the last task is to update the view window's display with another call to UpdateData().

8.5 Sorting and Filtering

In many cases when you're accessing a database, you want to change the order in which the records are presented, or you may even want to search for records that fit certain criteria. MFC's ODBC database classes feature member functions that enable you to sort a set of records on any field. You can also call member functions to limit the records displayed to those whose fields contain given information, such as a specific name or ID. This latter operation is called *filtering*.

8.5.1 OnSort function.

```
void CPACView::OnSortName()
{
      m_pSet->Close();
      m_pSet->m_strSort = "name";
      m_pSet->Open();
      UpdateData(FALSE);
}
```

```
void CPACView::OnSortPakno ()
{
        m_pSet->Close();
        m_pSet->m_strSort = "pakno";
        m_pSet->Open();
        UpdateData(FALSE);
}
```

All the sorting functions have the same structure. They close the recordset, set its m_strSort member variable, open it again, and then call UpdateData() to refresh the view with the values from the newly sorted recordset. A CRecordset object (or any object of a class derived from CRecordset, such as this program's CEmployeeSet object) uses a special string, called m_strSort, to determine how the records should be sorted. When the recordset is being created, the object checks this string and sorts the records accordingly.

8.5.2 The Filter Function

```
void CEmployeeView::OnFiltername()
{
        DoFilter("name");
}
void CEmployeeView::OnFilterpakno()
{
        DoFilter("pakno");
}
void CEmployeeView::OnFilterName()
{
        DoFilter("rank");
}


void CEmployeeView::DoFilter(CString col)
{
    CFilterDlg dlg;
    int result = dlg.DoModal();

    if (result == IDOK)
    {
        CString str = col + " = `" + dlg.m_filterValue + "`";
        m_pSet->Close();
        m_pSet->m_strFilter = str;
        m_pSet->Open();
        int recCount = m_pSet->GetRecordCount();

        if (recCount == 0)
        {
           MessageBox("No matching records.");
           m_pSet->Close();
           m_pSet->m_strFilter = "";
           m_pSet->Open();
        }
        UpdateData(FALSE);
    }
}
```

Whenever the user selects a command from the Filter menu, the framework calls the appropriate member function, either OnFilterDept(), OnFilterID(), OnFilterName(), or OnFilterRate(). Each of these functions does nothing more than call the local member function DoFilter() with a string representing the field on which to filter.

DoFilter() displays the same dialog box, no matter which filter menu item was chosen, by creating an instance of the dialog box class and calling its DoModal() function.If result doesn't equal IDOK, the user must have clicked Cancel: The entire if statement is skipped, and the DoFilter() function does nothing but return.

Inside the if statement, the function first creates the string that will be used to filter the database. Just as you set a string to sort the database, so, too, do you set a string to filter the database. In this case, the string is called m_strFilter. The string you use to filter the database must be in a form like this:

ColumnID = *`ColumnValue'*

The column ID was provided to DoFilter() as a CString parameter, and the value was provided by the user. If, for example, the user chooses to filter by name and types Aslam in the filter value box, DoFilter() would set str to Name = `Aslam'. With the string constructed, the program is ready to filter the database. As with sorting, the recordset must first be closed; then DoFilter() sets the recordset's filter string and reopens the recordset.

The DoFilter() function handles this by obtaining the number of records in the new recordset and comparing them to zero. If the recordset is empty, the program displays a message box telling the user of the problem. Then the program closes the recordset, resets the filter string to an empty string, and reopens the recordset. This restores the recordset to include all the records in the Employees table.Finally, whether the filter resulted in a subset of records or the recordset had to be restored, the program must redisplay the data--by calling UpdateData().

CHAPTER 9

ANALYSIS AND ACHIEVMENTS

Smart cards bring security, convenience and ease-of-use to millions of people worldwide. Smart cards find their place in a wide and growing range of applications from mobile telephony to CRM, loyalty, and to the development of next generation applications and services. With broad industry acceptance witnessed by increasing demand, businesses and service providers are constantly on the look for innovative smart card based solutions. Smart cards are proving to be of great value in applications like Electronic cash, Security, Access control, Transportation and RFID / IC Tags. Smart card area of concern are Security solutions for electronic banking platform using 3DES and PKI cryptography, ensuring secured financial transactions, Security solutions for mobile payments using SIM/WIM cards, Campus card solutions used to store and manage identification, access control, attendance monitoring, marks card and e-cash based payment information, Security solutions using smart card as a copy protection device for software, Solutions for distributors and retailers of consumer goods, Solutions for vehicle dealers to keep track of all services and repairs done for a vehicle, along with loyalty points accrued.

The smart card is uses the same evolution as PC but with a couple of years of delay (e.g. first commercial smart cards with 32-bit core are available only since very few years).The smart card is now moving from a close manufacturer dependant OS to an open card multi-applicative system like Java Card. To increase OS performances as well as cryptographic applications ones, smart card CPU are more and more designed in a way to improve capabilities directly linked to smart cards: new cryptographic instructions are introduced to improve cryptographic flexibility and reduce the importance of crypto-co-processors. The same work is done for Java dedicated instructions.The smart card is more and more secure, all the manufacturers develop and implement state-of-the-art countermeasures on all known attacks both in software and in hardware.

The simplest authentication application is really the classic ID card where the scope of use seems almost unbounded. There are really three underlying mechanisms:

Cardholder Authentication, Card Authentication,Transaction Authentication.Card authentication is relatively straightforward in that the card can contain a unique cryptographic identity that can be built into local and remote authentication processes. Limiting the cardholder to the card has always been the more elusive problem. The use of a PIN (Personal Identification Number) is well known but there still remains the problem of secure terminals that can pass the PIN into the card for checking. The only one you really trust is the terminal that you totally control. The use of biometrics is often touted as the saviour to this problem but an untrusted terminal could just as easily replay a biometric measurement as a PIN. It does however help the problem of impersonation if you can overcome the error problem. If you make the biometric control too rigid then you end up rejecting the true owner. If you are too generous on the controls then you make impersonation too easy.

There are two new application areas where the Smart Card is viewed to play a major role, access to computer workstations and LANs (Local Area Networks), and access to the Internet. Whilst these might still appear to be relatively small, being measured in a few tens of millions, there is no doubt that it will become far larger. It is in this area that Microsoft focused when they announced their Windows Card earlier this year. In classification user can differentiate between ID cards with local storage and ID cards with remote data storage. Looking at the basic application areas user can see driving licence and passports storing data on the card for local use whereas the access card for the Internet, as referred to previously, will be largely concerned with the management of remote data resources.

If user consider conventional credit/debit card instruments the user is really providing an authentication mechanism for accessing remote accounts. The card itself provides the account identification but the main accounting database is held remotely. If you look at Visa and MasterCard there are today something approaching 1bn cards in circulation worldwide. The use of Smart Cards for these payment instruments does, however, offer an additional security service and that is the ability to authenticate instructions, in this case a particular payment transaction. The fact that user has included all these applications in one area represents the view that in most cases an identity card has little value if it cannot authenticate instructions whatever the particular application may be. In other words a passive identity card that just stores identification data has little commercial value, it is nearly always necessary to challenge the card and owner to prove their identity and to be able to show convincingly, after the event, that the card was truly used.

CHAPTER 10

CONCLUSION & FUTURE WORK

10.1 Conclusion

The smart card is now moving from a close manufacturer dependant OS to an open card multi-applicative system like Java Card. To increase OS performances as well as cryptographic applications ones, smart card CPU are more and more designed in a way to improve capabilities directly linked to smart cards: new cryptographic instructions are introduced to improve cryptographic flexibility and reduce the importance of crypto-co-processors. The same work is done for Java dedicated instructions. The smart card is more and more secure, all the manufacturers develop and implement state-of-the-art counter- measures on all known attacks both in software and in hardware.

The project aimed at developing and implementing the universal architecture for smart card i.e. a single card would be used for multi-applications. The basic requirements were Smart Card Reader API Development, Smart Card Personalization, Implementation of Secure Smart Card Coding Scheme, Card & Database Interface Development, Integrating Universal Architecture, Remote Monitoring through sockets, Image Loading through Smart Card Handles were successfully accomplished. Project Deliverables were Universal smart card personalization software, Issuer stage smart card management software, Smart card access control management system, Smart card time and attendance management system, Databases in sql server, Smart card library management system, Smart card employee payroll management system, and Remote monitoring server.

Smart cards offer the ability to store secrets and personal data in a secure way, and to interact within a system using special communication interfaces and applications dedicated to specific protocols. This leads to a variety of services proposed by the card, services ranging from data management to hardware drivers, passing through crypto- graphic services. Applications for smart cards are numerous: financial services, mobile communications, health and transport services, e-business. An explanation for this success story lies in the symbiosis between hardware and software that is inherent to smart cards.

Sensitive data, such as personal data, secrets keys, and some application information have to be stored in memories. This write/store operation is far more protected than in any other devices. First the hardware will has special on-board security sensors, known VCC/Clock glitches detectors, to prevent the memory to be altered during data storage or reading. Then the software demonstrates backup mechanism in case of card power-down during storage. At the best all, access mechanisms can be a combination of usual Operating Systems (OS) access management added to systematic cryptographic verification (authentication, confidentiality), enhanced with hardware features that ensured hardwired “firewall” between memory areas. Eventually hardware/software protections exist against illegal tape-out reading. Enabling authentication and ensuring confidentiality and integrity imply that smart cards have the additional required capacity for arithmetic computations.

10.2 Future Work

The project aims at developing the universal smart card frame work for multi-application softwares. Many of the smart card application like secure access control management system, time and attendance management system and smart card library management system have been developed keeping in view the universal frame work of smart cards.. The project has glorious future prospects and few other applications that can be developed are described in this literature.

Smart cards have the power to revolutionize electronic toll collection (ETC). This revolution will not come about because smart cards are faster than current Electronic Toll and Traffic Management (ETTM) transponders. It will occur because smart cards offer toll agencies something no other technology can provide—an opportunity to get out of the banking business.

Cash is notoriously costly to handle. Current ETTM transponders eliminate cash at the toll booth but generate considerable costs of their own. The toll agency (or its vendor) must open one or more "tag stores," where motorists can purchase or lease transponders. Customers must establish accounts and pre-pay them by check or through automatic deduction from a bank account. The toll agency must provide customer service in person and by phone to handle problems involving transponders and accounts. In short, turning that split-second blip of transponder data into money in the bank requires a sizable back-office operation.

With smart cards, the back office disappears. No counting of bills and coins, no "shrinkage," no armored vehicles, no prepaid accounts, no back-office operation—the money flows from the customer's smart card to the toll agency's bank account as if by magic. Magic, of course, has nothing to do with it. Smart cards are simply the latest—and, one could argue, most convenient—means of electronic payment.

Smart cards can also provide a secure and handy medium for vehicle management systems in areas like, As driver licenses, As vehicle credentials, As payment media.

Suppose the driver license were carried on a smart card. The front and back would look as they do now—photo and demographic information on one side, notations and codes in excruciatingly tiny print on the other. Outwardly, no difference. Inside, Another matter entirely. The microchip inside smart card would carry not only name, address and physical description; it could hold your photographic likeness—compressed and digitized—and, perhaps, a fingerprint or other biometric measurement that is uniquely yours. Moreover, all that data would be protected by an encryption algorithm and secret key built into the microchip.

Think of the smart card as a computer that can process and store person’s medical record, and will grasp its potential power. Automated patient information would obviously speed up the processing of claims. Its greater value lies in preventing the sort of errors that routinely occur when physicians prescribe a drug or procedure based on a patient's symptoms but without access to key elements of the individual's medical and medication histories. The result in some instances is a drug or treatment that harms rather than heals. Adverse drug reactions take their heaviest toll among the very young and very old.

Many adverse drug reactions could be prevented if the physician who prescribed the drug and the pharmacist who dispensed it had better information about the patient. But all too often there is no patient "record." There are only fragments of information—some on paper, others in computers—scattered among clinics, hospitals and pharmacies.

That fragmentation inevitably leads to higher costs when examinations, laboratory tests and prescriptions are duplicated because one clinician has no access to the information others have gathered.

While many pharmacies routinely use computerized drug-screening software, physicians seldom do. The problem is not the software itself, which is available from multiple sources and reasonably priced. The problem is the fragmented nature of medical record-keeping. The best drug-screening software is of little value without a reliable patient record.

Suppose, instead, every patient came through the door with a relatively complete medical profile and insurance information. The physician would immediately be able to see what diseases, allergies and drug sensitivities had been diagnosed and what drugs the patient had been taking. For children, the card would document the immunizations that are so critical to the first few years of life.

REFERENCES

[1] M.-L. Akkar and C. Giraud. An implementation of DES and AES, secure against some attacks. In Pre- Proc. of CHES2001 - Workshop on Cryptographic Hardware and Embedded Systems, Paris, France

[2] D. Boneh, DeMillo R.A. and R.J. Lipton. On the importance of eliminating errors in cryptographic computations.

[3] I.F. Blake, G. Seroussi and N.P. Smart. Elliptic curves in cryptography. In London Mathematical Society Lecture Notes Series.

[4] Z. Chen. Java Card Technology for smart Cards: Architecture and Programmer's Guide. Addison Wesley.

[5] J.-F. Dhem, F. Koeune, P.-A. Leroux, P. Mestr ´e, J.- J. Quisquater and J.-L. Willems. A practical imple- mentation of the timing attack. In Proc. CARDIS'98, Smart Card Research and Applications, Louvain- la-Neuve, Belgium.

[6] J.-F. Dhem and J.-J. Quisquater. Recent results on modular multiplications for smart cards.

[7] K. Gandolfi, C. Mourtel and F. Olivier. Electro- magnetic analysis: concrete results. In Pre-Proc. of CHES2001 - Workshop on Cryptographic Hardware and Embedded Systems, Paris, France.

[8] H. Handschuh and P. Paillier. Smart card crypto- coprocessors for public-key cryptography.

[9] P.L. Montgomery.Modular multiplication with- out trial division.Mathematics of Computation,

[10] D. Naccache and M'Ra¨ıhi D. Cryptographic smart cards. IEEE Micro, pp. 14–24.

[11] W. Rankl and W. Effing. Smart Card Handbook,2nd Ed. John Wiley & Sons, 2000.

[12] R.L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems.

[13]Advanced Cards Systems
http://www.acs.com.hk
[14] Reinert, Lawrence A, Luther, Stephen C (1997), *User Authentication Techniques Using Public Key Certificates. Part 3: An Example Implementation*, National Security Agency, Department of Defense, USA.

[15] Reinert, Lawrence A, Luther, Stephen *Authentication Protocols for Smart Cards*, National Security Agency, Department of Defense, USA.

[16] Gerck, E (1997), *Overview of certification Systems*002C.
URL:http://www.mcg.org.br/cert.htm

[17] Baskerville, R (1998), *Security and Privacy of Information and Information Systems*, [Online accessed 8th April 1999].
URL: http://cis.gsu.edu/~rbaskerv/cis8680/index.html

[18] Pfleeger, Charles P (1997), *Security in computing, Prentice Hall*, Upper Saddle River, NJ.