Implementation of SCOSTA-CL based Smart Card Operating System (SCSOS)

by

Deepak Nagawade

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY, KANPUR

June 2008

IMPLEMENTATION OF SCOSTA-CL BASED SMART CARD OPERATING SYSTEM (SCSOS)

A Thesis Submitted

in Partial Fulfillment of the Requirements

for the Degree of

Master of Technology

by

DEEPAK NAGAWADE

to the

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY,KANPUR

June 2008

Certificate

This is to certify that the work contained in the thesis titled "Implementation of SCOSTA-CL based Smart Card Operating System (SCSOS)", by *Deepak Nagawade*, has been carried out under my supervision and that this work has not been submitted elsewhere for a degree.

June, 2008

_____

(Prof. Dr. Rajat Moona)

Department of Computer Science & Engineering,

Indian Institute of Technology Kanpur

Abstract

Smart cards are used extensively for data security and authentication purposes in various id card based applications. The data security in smart card is provided by access control based on passwords and cryptographic techniques. The smart card chip carries an operating system which implements cryptography and provides secure storage of data for application for interacting with the smart card. IIT Kanpur along with Government of India developed an application-independent smart card operating system standard known as SCOSTA-CL. SCOSTA-CL provides inter-operable command set for interaction between smart card data and applications.

In this thesis, we focus on the implementation of a SCOSTA-CL based Smart Card Operating System known as (SCSOS). The SCSOS implements contact and contact-less mode of communication with the smart card reader. The communication can be protected against eavesdropping through secure messaging. SCSOS implements symmetric key cryptography and has its own security architecture for the protection of resources such as data and commands. Along with resource protection, it also allows applications to perform cryptographic operations such as encryption, decryption, checksum computation and verification etc.

The SCSOS has been developed in a generic manner so as to support multiple applications on a single smart card. The modular design of SCSOS allows development and integration of new features with little effort. The developed operating system has been ported to the various existing smart card processors and rigorously tested for reliability and robustness.

Acknowledgments

I take this opportunity to express my sincere gratitude towards my supervisor Prof. Dr. Rajat Moona for his valuable guidance. I am grateful to him for his constant motivation, support and ideas that he gave throughout the year for my thesis. This work would not have been possible without his guidance. The flexible meeting hours with sir even on weekends really helped in the continuous progress of work. Overall, it has been great learning experience and I consider myself extremely fortunate that I got a chance to work under his guidance.

I would like to thank Ministry of Information System, Government of India for sponsoring this project. I would like to thank my colleague Aditi Gupta for the discussion that we had during my thesis work.

I would also like to thank Srishti Srivastava who helped me during the testing phase of my project work. I thank my classmates and friends for helping me during writing my thesis especially Anirudh, Vishal and Riyaz. I thank CSE faculty and department for their support to provide all the necessary resources required for my thesis. I immensely thank my family for their moral support during my work.

Contents

List of Figures

Chapter 1

Introduction

Smart cards were developed as an alternative to shortcomings of magnetic cards. The primary aim of magnetic cards is to provide machine readable data. However the magnetic cards provide no security against tampering. Further the storage capacity of magnetic cards is also very low and insufficient for many applications. This led to the demand of tiny device with a capability of larger data storage and better security logic. This need was fulfilled by the progress in the field of microelectronics in 1970s, which made it possible to integrate larger data storage and better security logic on a single chip. This chip also provides data security against unauthorized access and modification of data. Such kind of chips when integrated in plastic are known as smart cards.

The smart card was first introduced by German scientist Helmut Grottrup and Jurgen Dethloff [43] in 1968. The major progress of smart cards started when Roland Moreno registered his patent of smart card in France in 1974 [23]. Later in 1984, French postal and telecommunication agency successfully carried out field trial with telephone cards. In the field trial, smart card proved its reliability and protection against manipulation. By the 1990s, the smart cards were extensively used in France with the number rising to as much as 60 million cards in the France alone. Germany too followed similar footsteps. Smart cards were later introduced as bank cards in France in 1984. In 1994 smart cards came into the use of payment application by joint efforts of Master card [8] and Visa [14] together. Currently smart cards are used in many countries for various applications including passport applications [33], health care applications [23, 24], institute identity cards, payment systems [20], Telecommunication

Applications [22], Financial Transaction [35] etc.

The Smart card is defined as plastic card with integrated circuit embedded inside it with capability of transmitting, storing and processing of data. Smart cards are divided into three groups – memory cards, microprocessor cards and contact-less cards.

- Memory cards: These cards were first used in telephone applications. Later on they got popularity in transport [52], vending machines applications. The simple technology and low cost of these cards are the major advantages of these cards which give popularity to these cards in many applications. One of the disadvantages of these cards is that they can't be reused and recharged. Once a card is empty, it must be discarded. The basic purpose of these cards is to protect unauthorized read and write access to the data and provide some basic security related operations. This purpose is fulfilled by a simple security mechanism. Since these types of cards have simple technologies with very basic security logic, the cost associated with these cards is low.

- Microprocessor cards: These cards were first used in the bank cards in France. Apart from bank cards, these cards can be used in identification, secure data storage and electronic signatures applications. These types of cards have processor inside it for carrying out various security related operations and have large storage capacity compared to the memory cards.

- Contact-less smart cards: In these cards, data and power is transferred without any electrical contact between the card and the reader. The typical distance between the card and the reader for communication is around 10 to 15 cms. The power required for the operation of the card is generated with the help of a radio frequency (RF) field. Currently, the memory cards and the microprocessor cards both are available in the form of contact-less cards. Contact-less cards are used in variety of applications like which includes passport [33], ticketing [20], identification cards of company etc.

In this thesis, we discuss the microprocessor cards. We use the term smart card to specifically refer to microprocessor based smart cards.

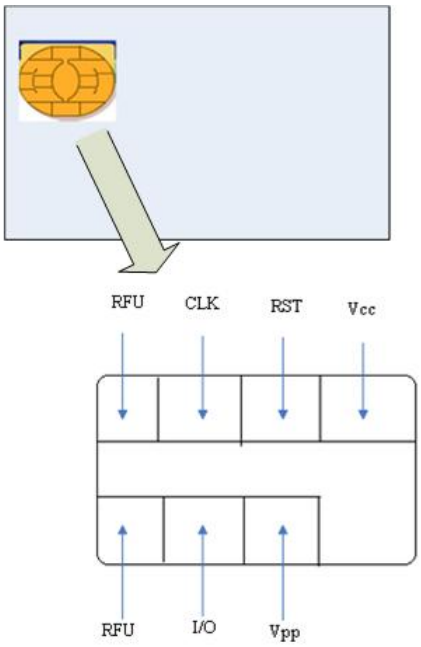

Figure 1.1: Physical Layout of Smart Card

1.1 Physical Layout and Components of Smart Card

Smart card is a plastic card chip with a chip module embedded inside it, as shown in the figure 1.1. This module has 8 contact pads on it. The physical interface of the card, with all its contact pads is shown in the figure 1.1. The contact pads are used for different purposes.

- The I/O contact pad is used for communication of data from the card to the reader. Since, there is only one contact pad for data transfer, the communication with the card is half duplex. At a time the card can either receive data or send data to the terminal. Terminal is the device which exchanges information with the card.
- The Vcc pad is used to supply external power to the card.
- The GND pad is connected to the ground of the external terminal.
- The RST pad is used by the terminal to send the reset signal to the card. At any point of time card can be reset by the terminal using this pad.

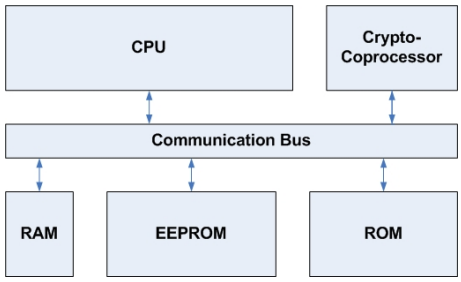

Figure 1.2: Components of Smart card

- The Vpp pad is used for supplying programming voltage to the card. The additional voltage may be needed for the programming of data onto the EEPROM.
- CLK pad is used to transmit the clock signal to the card.

A smart card has various components inside the chip. These include CPU, EEPROM, ROM, RAM, communication interface etc. as shown in figure 1.2. The ROM contains the card's operating system. This operating system uses the EEPROM for storing and retrieving data in the smart card organized as files. The RAM is the volatile working memory used by the programs running on the CPU. Most smart cards chips today have 8 – bit micro-controller with varied EEPROM sizes around the range of 4KB to 144KB, ROM sizes around the range of 4KB to 200KB, and RAM sizes around the range of 256 bytes to 6KB. The configuration of various components of the card depends upon type of the card, cost of the card, application requirements etc.

1.2 Smart Card Operating System

The smart card processor executes a software that provides a standard interface. This software is known as operating system. The major functionality of the operating system is to provide a standard way of information interchange between the card and the terminal device, or a computer, through smart card interface device commonly known as smart card reader. The communication between the smart card reader and the smart card is carried out in form of commands and their responses as follows.

1. The smart card reader sends a command to the card operating system.
2. The smart card operating system receives command via serial I/O interface.

3. The card operating system executes the command and sends out a response that includes the status code to indicate errors if any.

Apart from the information exchange and the execution of commands, smart card operating system also provides security of data against unauthorized access and file management.

1.3 SCOSTA-CL and ISO Standards

IIT Kanpur in association with the Govt. of India defined an operating system standard for smart cards known as SCOSTA [13]. SCOSTA is ISO compliant and application independent standard. Originally this standard was developed for travel applications such as driving licenses [53], vehicle registration [18] etc. The commands and the operations supported by this standard can be used for other applications as well. SCOSTA supports symmetric key cryptography and basic security related operations. A recent security enhancement was carried out in this standard to support protection against eavesdropping with the communication using secure messaging. This feature is essential for contact-less operations. This version of standard is known as SCOSTA-CL [31]. The SCOSTA-CL standard provides commands suitable for contact-less operations with some additional security related features for the smart card operating system. This extended version of SCOSTA standard is compliant with ISO as well as International Civil Aviation Organization (ICAO) [33] standards for electronic passport.

ISO standards for smart cards describe various aspects of the smart cards. These standards define the physical layout, the electrical interface, the communication parameters, the communication protocols of the card and the command structure as well as interpretations. ISO 7816-3 [39] defines communication protocol between the smart card and the smart card reader. ISO-7816-4, 8 and 9 [38, 41, 40] define the commands supported by the smart card including the various security related commands. ISO 14443-3 [34] and ISO 14443-4 [37] deal with the contact-less data communication protocols.

1.4 Objectives of this Work

The objective of this thesis is to design and implement an operating system based upon SCOSTA-CL standard for smart card. There are the following requirements for developing operating system.

1. The operating system should be easy to port on several smart card processors. Due to this requirement the design of the operating system should be modular and must be organized in processor dependent and processor independent manner.

2. The operating system must support proper file system structure in order to provide functionality of embedding more than one application on a single smart card. In addition to this the operating system must support anti-tearing mechanism to ensure that the command is executed atomically (i.e. effect of execution is never partial).

3. The requirement for passport application is to be considered while designing the OS. This application requires fast and secure reading of data. Therefore security related operations like encryption, computation and verification of checksum should get executed quickly with appropriate security mechanism in place. The speed of communication between the smart card and the reader should be high to reduce latencies in the communication.

4. The operating system should have separate infrastructure for supporting security related operations and should maintain the status related to keys and passwords to be authenticated.

1.5 Organization of Thesis

The rest of the report is organized as follows. In chapter 2, we describe the background of the system including the basic command and response structure for communication with the card, communication protocols, basic data structure file structure, security mechanisms and security environment of the card. In chapter 3, we describe design of system including file system of smart card, structure of the operating system, cache in operating system, anti-tearing mechanism and security infrastructure. In chapter 4, we describe the implementation including secure messaging support, generic security routines, managing security environment, contact-less protocol etc. In chapter 5, we describe portability

aspect of smart card such as handling of multiple Answer to Reset (ATR) and porting of operating system on different processors. In chapter 6, we describe various testing debugging tools used for the development. We also discuss test cases used in testing. Finally we conclude the thesis and describe the future enhancements in chapter 7.

Chapter 2

Background

As described in the previous chapter, the card operating system (OS) receives commands from the reader, interprets the commands, executes them and sends the responses. The basic unit used for the command and the response transmission is known as Application Protocol Data Unit (APDU).

2.1 APDU Structure

Commands and responses are transferred in the form of an APDU structure. The APDU used for transfer of the command is known as command APDU, while APDU used for sending the response is known as response APDU. The command APDU (figure 2.1) has two parts – command header and command body. The command header contains CLA, INS, P1 and P2 bytes, while the command body has Lc byte, command data field and a Le field.

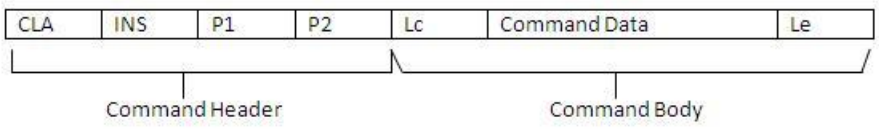

Figure 2.1: Command APDU

- CLA: This is the first byte in the command APDU. This byte encodes the class of the command, logical channel number etc. The class of the command as indicated by this byte can be a plain

command or the command with secure messaging [38].

- INS: This is the second byte of the command APDU and identifies the instruction to be executed.
- Command Parameters (P1 and P2): These command parameter bytes are related to the INS byte and further qualify the instruction.
- Lc byte and command data field: Lc byte provides the length of the command data field while command data field gives the data required for command execution. The presence of Lc byte and command data field depends upon the type of command. If the command is an input command or a bi-directional command, the Lc byte and command data field are present.
- Le field: The value of this field conveys the information about the maximum number of bytes the reader expects in the response field. This field may be empty for an input-only command.

There are four types of commands.

Type I: In this type of commands, command body and the response data are absent. A type I command APDU does not have Lc, command data field and Le field.

Type II: In this type of commands, Le field is present while Lc byte and command data field are absent. Such a command is an output-only command.

Type III: In this type of commands, Lc byte and command data field are present while Le field is absent. Such a command is an input-only command.

Type IV: In this type of commands, Lc byte, command data field and Le field are all present. Such a command is a bi-directional command.

The response of the command is sent using the response APDU and it has two parts, response field and status bytes, as shown in figure 2.2.

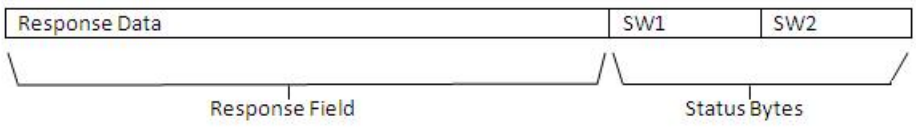

Figure 2.2: Response APDU

- Response Field: The card gives out the response to the command in the response field. The number of bytes present in the response field cannot exceed Le field as given by the reader in the command APDU.
- Status Bytes: These bytes convey the status of the command processing after command execution.

2.2 Communication Modes

There are two modes of communication between the reader and the card – contact mode and contactless mode.

2.2.1 Contact Mode

In the contact mode, the communication takes place through electrical connections between the reader and the card. T=0 and T=1 are the protocols used for the contact communication as defined in ISO 7816-3 [39].

T=0 is a byte oriented, half-duplex protocol, i.e., at one time either the card or the reader can transmit signals. When a reader sends the command then the card is in the receive mode. When the card sends the response the reader is in the receive mode. Since this is byte oriented protocol, command and response APDUs are transferred using communication handshake for each byte.

T=1 is a half-duplex block-oriented communication protocol. Therefore, at a time only one party can be in the sending mode and other one will be in the reception mode. The command and response APDUs are transferred using blocks. Some kinds of additional blocks are used for acknowledging the previously sent blocks and to control information exchange between the reader and the card. These additionally supported blocks apart from the blocks for APDUs transfer provide reliability in the communication.

2.2.2 Contact-less Mode

In the contact-less mode, the card is not in physical contact with the reader and radio frequency (RF) field is used for the communication. The card generates power using this RF field. The same RF Field is also used for exchange of APDUs between cards and the reader. The protocol for contact-less communication is defined in ISO 14443–3 [34] and ISO 14443–4 [37] and it is discussed in chapter 4 of this document. This protocol is half-duplex block-oriented transmission protocol.

2.3 Communication Parameters

Communication parameters are used by the OS for the communication with the reader. Some of these parameters are global in nature and affect the card operations in general. Other parameters are protocol specific ones. For example, parameters like programming voltage, programming current etc. are global parameters and are used by all protocols supported by the card. Parameters like data rate, block waiting time, character waiting time, frame size [39], etc. are specific to the T=1 protocol.

When the contact card is initialized by providing a power, the processor gets reset and provides an initial sequence of bytes indicating the card capabilities. This sequence is known as Answer To Reset or ATR. An ATR has information about protocols supported by the card and possible communication parameters for each of the protocol supported. The reader selects one of the protocols from the available protocols and negotiates the communication parameters for that protocol. This negotiation of communication parameters and selection of a protocol among the available protocol happens through handshake of bytes known as Protocol Parameter Selection (PPS) bytes. The PPS request can come immediately after the ATR is given out by the card. If the PPS request arrives after receiving the first command, the request is not processed. This means that it is not possible to change the protocol or its parameters in the middle of a session.

Communication parameters for the contact-less protocol are indicated by using the Answer To select (ATS). An ATS is a string that card returns to start the contact-less communication with the reader. The ATS has information about baud rate for transmission [39], maximum frame size that the card can receive etc. The data transmission rates can further be negotiated and selected by the PPS

data exchange. The PPS request is enabled after ATS is given out by the card and is disabled after the card receives the first command. Since, the PPS request is disabled, communication parameters cannot be changed during the session.

2.4 Communication of APDUs

The APDUs are transmitted using Transmission Protocol Data Units (TPDU). TPDUs define the bytes transported by the transmission protocols. They can be regarded as protocol dependent containers that transport data, to and from the card. The TPDUs used to carry to command are known as command TPDUs and TPDUs used to carry response are known response TPDUs.

In the case of T=0 protocol, an APDU needs to be converted to a T=0 protocol TPDU for the transmission. In this protocol, a TPDU can support data transfer only in one direction. Thus, a type IV command can not be carried using a single TPDU. Since it involves data transfer in both directions. Therefore in this protocol, type IV command is broken into two commands one of type II and another of type III and carried in two separate TPDUs.

In the case of T=1 protocol, the command and response APDUs are encapsulated in blocks. The blocks in this protocol are transferred as TPDUs. In this protocol, it is possible to transmit all types of commands in a single TPDU.

In the case of contact-less protocol, the command and response APDUs are encapsulated in blocks. In this protocol, frames are used for data exchange. Therefore, blocks are transferred in frames.

The block structure and TPDU transfer is discussed in ISO 7816-3 [39] standard for contact protocols and ISO 14443-3 [34] and ISO 14443-4 [37] for contact-less protocols.

2.5 Basic Data Structure

The data is stored in the form of files in the card. These files are arranged in a tree structured fashion with the root of the files known as Master File. Figure 2.3 shows the tree structure of the file system. The depth of the tree is restricted primarily because of the limited size of EEPROM where files are stored.

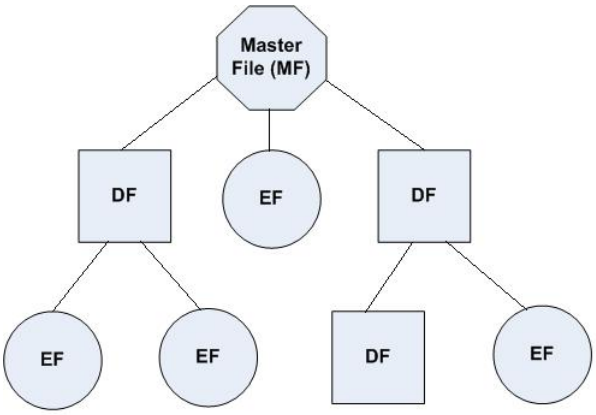

Figure 2.3: Logical File Organization

There are two categories of the files, elementary files (EFs) and dedicated files (DFs), as shown in the figure 2.3. EFs store the data related to the applications. DFs are placeholders for the EFs and DFs in the directory tree structure. These DFs are used for identifying application if the OS supports multiple applications. The files stored under an application specific DF are used only for that application.

2.5.1 Elementary Files

The EF files are used to keep the data pertaining to application. An EF can be of following types.

- Transparent EF: These files contain unstructured data as a stream of bytes. Application can organize the data in whichever form needed.
- Linear EF with fixed length record: These are record files containing records fixed in size. The records in these files are linearly stored.
- Linear EF with variable size record size: These are record files containing records varying in sizes. The records are linearly stored.
- Cyclic EF with fixed size of records: The fixed length records in these types of EFs are logically stored circularly. Thus when a record is added to the file, the oldest record is deleted if there is not enough space in the file.

2.5.2 File Referencing Methods

Following are the methods to refer to files stored in the card.

- Referencing by file identifier: The EFs and DFs can be selected by using two byte file identifiers. This method of referencing with a file identifier is applicable to the currently selected DF. Therefore, to select a file uniquely, the EFs and DFs under currently selected DF should have unique file identifiers.

- Referencing by path: A file (EF or DF) can be selected by using path. The path starts with file identifier of the MF, or file identifier of the currently selected DF, and ends with the file identifier of the file to be selected. In between these two identifiers, the path consists of zero or more file ids of intervening DFs.

- Referencing by short identifier: EFs have an optional short file identifiers (SFIDs) associated with them. An EF under the currently selected DF may be selected implicitly using its SFID in various commands. However, SFIDs cannot be used in the path and are not the substitute for the regular file identifier.

- Referencing by DF name: DFs can have optional names associated with them. It is possible to select DF by its name. In order to select DF by name, the DF names should be unique across all DFs in the file system.

2.5.3 Data Referencing Methods

Data stored in the card can be referenced as records, data units or data objects. The data is stored as a sequence of records in the record files or as a sequence of bytes (data-units) in transparent files. In case of transparent EFs, data can be accessed by giving offset of the first byte from the beginning and the number of bytes to be read. While in case of record oriented files, records can be accessed using record identifier or record number.

- Record referencing by record identifier: The record identifier is provided by the application at the time of creating a record. Within an EF, record identifier can be same for two different

records.

- Record referencing by record number: The record numbers are unique and sequential within an EF. The numbers are assigned to record while writing in case of files with linear structure. While the record numbers are sequentially assigned in the opposite order in case of cyclic files. i.e. The record with record number 1 is most recently created record.

Along with data storage in the form of records, ISO standards provide variety of ways to organize data in tagged format. Data is organized as 3-tuple with members tag, length and value (TLV) known as data objects and it is encoded in simple TLV or using Basic Encoding Rules (BER) [50]. In simple TLV form, tag is 1 byte, length is 1 byte and no nesting of DOs is done while BER is coded using ASN.1 representation [50].

There are two types of data objects (DOs) – primitive and constructed. In constructed DOs several primitive or constructed DOs can be nested in a constructed DO while in case of primitive DOs no nesting is done.

The DOs can be stored under MF or DF in the smart card. The EF in the smart card can contain primitive as well as constructed DOs for storage of FCP of a file, encoding of file identifier, etc.

2.5.4 File Control Parameters (FCP)

Control information of a file stored inside the card is associated with the file at the time of file creation and is known as a collection of file control parameters (FCP). These file control parameters include file identifier, short file identifier, DF name, file descriptor byte (FDB) that specifies the type of the file, number of records, size of records etc.

2.6 Security Mechanism

The various resources in the card such as files, data objects etc. are protected against unauthorized access by using the access control mechanism. The access control mechanism for files and DOs is provided in the form of compact and expanded attributes [40]. These attributes specifies the security conditions that must be satisfied to allow access to the files and DOs. The external entity accessing

the card authenticates itself by using some security mechanisms for authentication. Following are the authentication mechanisms supported by the card.

- Entity authentication based on password: In the password based authentication, the password entered by the user is verified against the stored password in the card.
- Entity authentication based on key: Key based authentication is based upon challenge and response mechanism. The challenge is given to the party to be authenticated, which operates on the challenge using symmetric key and provides a response to the challenge issuing party. The challenge issuing party verifies the response against previously issued challenge for successful authentication. For example, a response may be formed by encrypting the challenge. Key based authentication includes internal authentication, external authentication and mutual authentication. In the internal authentication, the card is authenticated with the reader to ensure genuineness of the card while in external authentication, the reader along with the application is authenticated to the card. In mutual authentication, internal as well as external authentication takes place simultaneously, i.e., card and reader authenticate each other.
- Data authentication: This includes computation and verification of cryptographic checksum on the data. The checksum is computed in order to provide integrity of data so that any modification in the data can be detected. The procedure of checksum computation and verification requires a key.

The entity authentication mechanisms described above are used for checking the security condition. The access to the files and DOs is granted, if the security conditions are met. In addition to above authentication mechanism, the card supports other security mechanisms including data confidentiality, integrity and command confidentiality and integrity using secure messaging.

- Data confidentiality: This mechanism provides confidentiality by data encryption and decryption so that only the genuine key holders are able to see data.
- Data integrity: This mechanism deals with the computation of message authentication code (MAC) or hash. For different messages it is almost impossible to generate same MAC hence it is used to detect alteration in messages.

- Secure messaging: This mechanism is used for secure transmission of commands and responses. This secure transmission is achieved by confidentiality and integrity over command and response APDU.

2.7 Security Environment

A security environment (SE) is a logical container having security components required either for the security mechanism of the card or for secure messaging. The SE is identified using security environment identifier and it contains control reference templates (CRTs). The CRT is set of control reference data objects (CRDOs) which have the information about parameters such as key, the algorithm, the initial value (IV) etc. in the form of DOs. These CRTs are used for operations such as authentication, encryption, and checksum computation etc. Following are the types of CRTs and operations associated with them.

- Confidentiality template (CT): This template has CRDOs for the confidentiality related operations such as encryption, decryption etc.
- Crypto-check-sum template (CCT): This template has CRDOs for the computation and verification of check-sum.
- Authentication template (AT): This template has CRDOs for internal authentication, external authentication and mutual authentication operations.
- Hash template (HT): This template has CRDOs for hash computation.
- Digital signature template (DST): This template has CRDOs for digital signature calculation.

The operations for which a CRT is defined is controlled by a CRT usage qualifier. The CRT usage qualifier specifies the operation for which the parameters specified in a particular CRT are to be used. Each of the CRT may or may not have CRT usage qualifier associated with them. If the CRT usage qualifier is not available then the default value is taken as specified in SCOSTA-CL document.

The card always has a default SE which contains templates for performing various security related operations such as encryption, decryption, check-sum computation and verification. This default SE is

known as current SE and it can be empty (containing no templates). Parameter for security operations are taken from the current SE unless some other SE is explicitly specified.

2.8 Related Work

One of the smart card operating system was STARCOS [29] developed by Giesecke and Devrient in 1990s. It supports multiple applications on a single smart card. TCOS [25] is another OS which can be used for different applications and is compliant to ISO 7816 standards. Multi-application payment card operating system (MPCOS) [44] from Gemplus is the microprocessor card introduced in 1990s and used in applications for government, health insurance, banks and public organizations. As the name indicates, this OS can be used for multiple applications. The operating system for smart card is developed in open source too. Simple Operating System for Smartcard Education (SOSSE) [12] is one of the open source operating system for smart cards.

Europay [3], Mastercard [8] and Visa [14] jointly developed EMV standard [25] for credit cards containing smart card chips. The widely used implementations of this standard include VSDC [55] from Visa and MChip from MasterCard. The specifications of this standard state that a tree structure must be used for files. The credit card application in the card is located in its own DF, which contains all data elements required for credit card application. These data elements are stored in EFs with standard file structures in accordance with ISO 7816-4 [38]. EMV [20] uses static unilateral authentication with card specific keys. This OS does not allow card to authenticate the reader. The authentication is not essential in EMV applications because debit operation is not performed from the card. The cryptographic algorithm used in EMV for authentication is RSA algorithm.

The SCOSTA [32] standard for smart card OS defines the commands and data structure for the card. The files in SCOSTA are arranged in a tree structured form, using DFs and EFs. The master file (MF) is the root of the tree and EFs and DFs are created under MF. This standard supports several types of EF, data and file referencing methods described earlier in this chapter. It has support for entity authentication based upon passwords and keys, data authentication, encipherment, decipherment and cryptographic checksum computation. This standard has support for CT, CCT and AT templates which have CRDOs for encryption, decryption, computation and verification of checksum and for various

authentication purposes. SCOSTA supports symmetric key cryptography through TDES algorithm. The SCOSTA standard was released in public by the Government of India. Several implementation of the smart card operating system based on this standard are available. Since the IIT Kanpur defined the SCOSTA specifications, it also had its own operating system implementation known as IITK-SCOSTA [54]. IITK-SCOSTA supports T=0 protocol for the communication in contact mode. It has support for anti-tearing mechanism which provides protection to card data against data corruption due to card tearing when the card is suddenly pulled out of the reader. The OS was implemented for Infineon SLE66 processor [45] and Linux as an application. The security module of this OS was developed by Ankit Jalote and Marghoob Mohiyuddin [42]. Ravinder Shankesi developed individual command handlers, support routines etc. [54]. This OS was later ported to NXP P8 family of processors.

ACOS5 [17] is a card OS developed by Advanced Card System (ACS) [1]. This OS supports only T=0 protocol. The file structure of ACOS5 is compliant to ISO 7816–4 file structure and files are arranged in tree structured form with root of all files as master file (MF). This OS has support for anti-tearing mechanism. ACOS5 supports both symmetric as well as asymmetric key cryptography. It supports AT, CCT, CT, DST and HT templates which deal with the corresponding security related operations. DST template specifically deals with only asymmetric key cryptography while other templates can be used for symmetric as well as asymmetric key cryptography. This OS has support for security mechanisms as described in section 2.6.

STARCOS [29] supports communication in dual mode, i.e., contact as well as contact-less mode. The protocols supported in contact mode are T=0 and T=1. The file structure of STARCOS is based upon a root directory with sub-directories and files. The master file (MF) is root directory of the file system and dedicated files are sub-directories of MF. The MF and DFs can contain elementary files (EFs), internal secret files (ISFs), and internal public files (IPFs). ISFs store private keys, personal identification numbers while IPFs store public keys. ISFs and IPFs are specific type of EF. This OS has support for both symmetric as well as asymmetric key cryptography. It has RSA support for digital signature generation and session key establishment. This OS supports security mechanisms as described in section 2.6.

Aforementioned operating systems have a fixed command set. While there are operating system

which allow the programmer to program card. The term open platform is used to refer such smart card operating system. MULTOS [21] is one of the OS in open platform. This OS is a multiapplication smart card operating system compliant to ISO/IES 7816-4. The program code is typically developed in 'C' and translated into MULTOS executable language (MEL) using a special compiler.

Java Card 1.0 [7] is another operating system in open platform which provides an application programming interface (API) for integrating Java in smart card operating system for its MF, DF and EFs to be accessible using Java. The cost of such generality is the extra processing capability required on the card for implementing the virtual machines for these languages. These cards are approximately three times more costlier than the ordinary cards. The first Java card was introduced in 1997 by several companies including former Schlumberger's card division (then Axalto) [10] and Gemplus (both merged in Gemalto) [5]. The security in these Java cards is provided by various aspects of Java technology as mentioned below.

- Applet Firewall: Different applications are separated from each other by an applet firewall which restricts and checks access of data elements of one applet to another. An applet is a program written in the Java programming language that can be included in an HTML page.
- Cryptography: Commonly used encryption algorithms like DES, 3DES, AES, RSA (including elliptic curve cryptography) can be supported using Java classes.

The Java card does not directly support file oriented structure. The file management, for example using an ISO 7816-4 file structure [38], must be specifically implemented in Java using several classes.

ZeitControl [16] developed BasicCard [2] which is the first smart card programmable in Basic. It allows user to program application in Basic and download them into the smart card. The primary advantage of BasicCard is that basic can be easily interpreted and no sophisticated mechanisms are used for preparing applications. These cards have support T=0 and T=1 data transmission protocols in contact mode, and ISO/IES 14443 type B [37] for contact-less data transmission protocol. It has support for normal cryptographic algorithms such as DES [28], Triple DES, AES [27], RSA etc.

In 1998, Microsoft [9] introduced windows smart cards (WSC) [15]. These cards are designed to be used for multiple applications and they support downlodable program code. The file management

of this OS is based upon ISO 7816-4 [38] and ISO 7816-9 [41]. The file management in this OS used file allocation table (FAT) which is the unique feature provided by these cards.

Keycord once marketed a smart card called OSSCA (Operating System for Smart Sard Applications) [12] which was programmable in Forth [4].

In addition to above mentioned operating system, GemXplore, GPK by Gemplus [5]; STARSIM, STARDC by Giesecke & Devrient [6]; CardOS by Siemens [11]; Cyberflex, Multiflex, Payflex by Schlumberger [10] are some of the smart card operating system marketed by different companies.

The smart card operating system described in this report is compliant to SCOSTA-CL [31] standard. It has a fixed set of commands and support for symmetric key cryptography.

Chapter 3

System Design

In this thesis, we implement a SCOSTA-CL based Smart Card Operating System (SCSOS). The SCSOS is designed with the following goals.

- Maintainability: The code for the SCSOS must be easy to maintain. It should allow flexibility to add new features in the SCSOS without major changes in the existing code structure. This is achieved by modular programming.
- Portability: The SCSOS is designed in such a way that it is possible to port it to various processors with little effort. For this purpose, the SCSOS code is divided into parts, processor dependent part (PDP) and processor independent part (PIP). This division ensures that only PDP has to be changed while porting SCSOS to a different processor. The efforts are made to keep the PDP as minimal as possible.
- Memory constraints: The smart card is a small device with limited amount of memory. Hence, the SCSOS must use this memory optimally. Various methods like buffer reuse, variable reuse etc. are implemented in the SCSOS for the optimal use of memory.
- Reusability: Common routines are reused in order to keep code size minimal. The modular approach for the SCSOS implementation enhances the reusability of the code.
- Generic operating system: The SCSOS is designed independent of the application that is going to use it.

- Efficiency: The SCSOS is designed for efficient use of the available resources.

3.1 Structure of SCSOS

SCSOS is divided into processor independent part (PIP) and processor dependent part (PDP). The PIP incorporate functionalities such as initialization, receiving command APDU, security check, command handling and composing a response as shown in figure 3.1. The PDP incorporates functionalities such as handling of EEPROM, transmission of APDUs, random number handling etc.

3.1.1 Initialization Module

This module initializes code variables and carries out some functionality which is necessary before the card is ready to receive command APDUs. This module is further divided into two modules, a module that handles the initialization of contact protocols and another module that handles initialization of contact-less protocol.

In the contact protocols, the card gives out an Answer To Reset (ATR) and may or may not receive the request for protocol parameter selection (PPS). If the PPS request is received then it is processed to set the protocol dependent parameters such as data rate, block waiting time [39] etc. Finally, the card gets ready to receive the command APDU.

In the contact-less protocol, the card gives out an Answer To Select (ATS) and may or may not receive PPS request. If the PPS request is received then it is processed to set the protocol dependent parameters such as baud rate. Finally, the card gets ready to receive the command APDU.

3.1.2 APDU Receiving Module

The APDU receiving module receives the command APDU from the reader. The receive operation is protocol dependent.

In T=0 protocol, TPDUs are sent between the reader and the card and they define the APDUs as per protocol encoding. The command TPDU in this protocol carries CLA, INS and three parameter bytes and an optional data field. If the command is input only or input/output type then the third

parameter byte give number of bytes to be received by the card. For output only commands the third parameter byte is used as Le field.

In T=1 protocol, the commands and responses are transmitted using T=1 protocol TPDUs. In this protocol, the TPDUs are transmitted as blocks. The error handling in block receive and transmit operation is done at the protocol level. Finally, the APDU is extracted from the received TPDU.

In contact-less protocol, frames are used to receive and to transmit the APDUs. Error handling in achieved is achieved at frame level. The APDU is encapsulated in these frames which are extracted by this module.

The APDUs may indicate presence of secure messaging. If secure messaging is used then the corresponding secure messaging data objects are processed, the errors are checked and the plain APDU is extracted. Thus, at the end of execution of this module the card has plain APDU from command processing. If the plain APDU is not received correctly as per command type and protocol then the card returns an error and command execution is aborted.

3.1.3 Security Check Module

The security check module is used for checking the command level and file level security for each and every command. For checking the command level security, it is checked whether the command execution is permitted or not based on the current security status and the security attributes of currently selected DF. The security attributes associated with the currently selected DF has control information in the form of expanded attributes which define the security environment to be satisfied to permit command execution.

For the file level security, permissions are checked to perform file level operations like read, write, and update etc. These permissions are checked by using the security attributes in the compact and the expanded form [40] for the file. The command is allowed to execute, if the security status of the card permits the corresponding operation. Otherwise, the command is aborted and appropriate security error is returned.

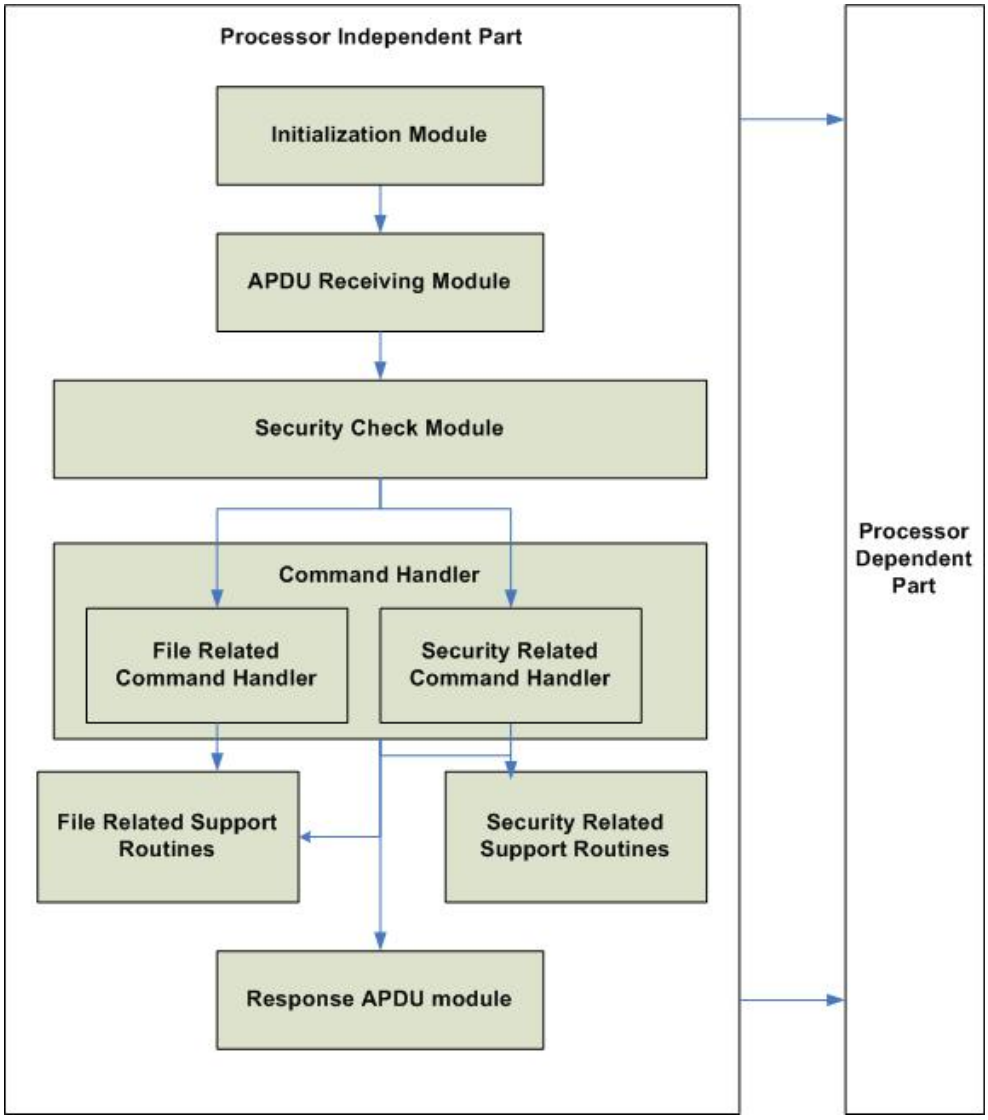

Figure 3.1: Modules of Operating System

3.1.4 Individual Command Handler

The individual command handlers are invoked for the processing of commands. The command handlers perform command execution and set the status bytes. The status bytes give the information about success or failure of the command. There are mainly two types of commands – file related commands and security related commands.

During the execution of file related commands the card status related to the currently selected DF, and EF, record related status such as current record etc., are set or modified. During the execution of security related commands, the security status of the card such as the passwords/keys that are authenticated is modified.

3.1.5 Support Routines

The PIC modules of SCSOS provide various support routines used by individual command handlers for the processing of the commands. The support routines include the methods to access meta-data information of the file, to write file bytes, to find free space in the file system, to navigate through the file system, to perform encryption and decryption, to compute and verify cryptographic checksum, to handle access conditions on the key etc.

3.1.6 Response APDU Module

The response APDU module sends the response along with the status bytes to the reader. The process is protocol dependent and handles the protocol errors while sending the response.

3.1.7 Processor Dependent Part

The processor specific part of the SCSOS include code for handling functionalities like reading from EEPROM, writing into EEPROM, generating random numbers, reading ATR, handling the contactless interface, handling the input/output etc. The PDP gives processor specific support to the PIP during the command execution.

The initialization of the system includes the processor specific initialization. This involves setting processor specific parameters such as control registers to select the clock of CPU and co-processors

[47], the communication parameters etc.

The PIP reads the command APDU and sends the response APDU. For this it needs the support of processor specific input and output.

The file system implementation needs the EEPROM read and write support. A processor specific efficient support is provided which can merge several read and write requests into a single page read and write request thereby increasing the efficiency.

The cryptographic operations such as encryption, decryption can be performed by specialized hardware if supported by the processor. The processors usually provide this functionality through crypto co-processors. The crypto co-processors perform cryptographic operations at a speed higher than the implementation in software. The DES, TDES algorithm implementations at hardware level has been added in the system.

3.2 File System of SCSOS

The smart card hardware typically provides only EEPROM as non-volatile storage. In SCSOS, EEPROM is used to store the file system as well as other non-volatile information. For this purpose, the EEPROM is divided into several partitions. The file system is implemented in one such partition of the EEPROM.

3.2.1 Logical File System

The logical layout of file system is as shown in figure 3.2. Every file has a link to its sibling and a link to its parent directory. The last file in the sibling chain has its sibling link as NULL. The parent link of the master file (MF) points to itself. The directory files (DFs) store a link to the first child.

Given the staring address of a DF, it is possible to traverse the children of the DF. The first child is followed using the child link of the DF. Remaining childrens are followed using the sibling link of the children. Thus, all children of DFs can be accessed using child and sibling links. Likewise, entire file system can be traversed starting from MF.

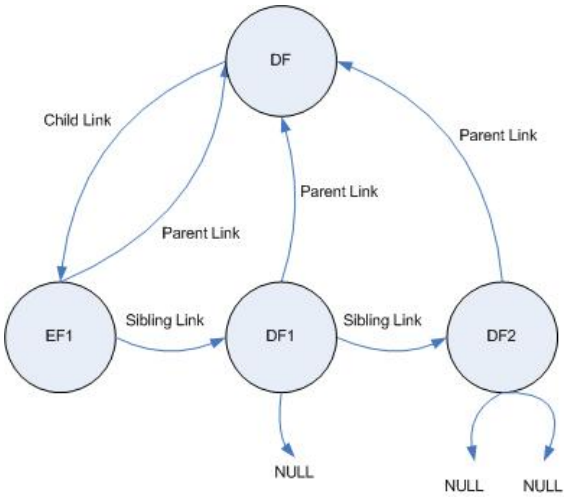

Figure 3.2: File System Layout

3.2.2 The Physical Layout

The file system stores the data and control information of files and DOs using contiguous allocation in the EEPROM. For this the data is organized in the form of blocks. There are three types of blocks in the EEPROM – free space block, DO block and file block. We look at the contents of each of the block.

- Free space block: These types of blocks are used for storing free space information. The first byte of this block is '0xFF' used as the "free space indicator". The next few bytes of this block give the length of the free space block. The free space blocks are used for the creation of files if the space requirements are fulfilled. We use first-fit method to allocate blocks for the files and DOs.

- File block: These types of blocks are used for storing files. The first byte of this block is the FDB of the file. The next few bytes of this block give the length of the block. The length of block is followed by meta-data or control information of the file and optional body containing data in the file. Data is absent in case of dedicated files. The meta-data of the file include pointers to sibling and parent, child pointer of the first child in case of a DF, FCP, bit vectors to support write once, write-and and write-or operations etc.

- DO Block: The first byte of this block is 0x80 used as "DO indicator byte". The DO block also

contains the length of the DO block, the parent pointer to the DF, the sibling pointer to other DO blocks, tag bytes of DO, length and value of the DO.

3.3 Cache in SCSOS

The SCSOS implements cache for anti-tearing mechanism. The data is first written into the cache and only after the state the state file system is stabilized, it is copied to the file system. This ensures that the data is updated automically to the file system. This cache is implemented in one partition of the EEPROM.

3.3.1 Cache Structure

The cache structure is shown in figure 3.3. The first byte of the cache marks the cache to be valid or invalid. This is followed by space to store cache length and cache blocks. The cache block structure has three parts to store the address in the file system, the length of the block and the data.

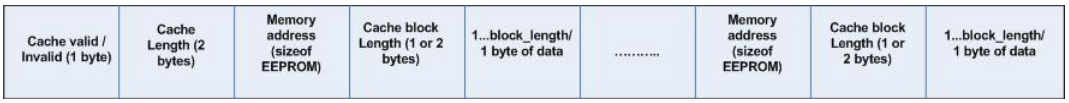

Figure 3.3: Cache Structure

The cache is set to invalid before the command execution starts. The data to be written into the file system is written to the cache during the execution of the command instead of writing into the file system. At the end of the command execution, if there is data in the cache, cache is validated. The contents of a validated cache are copied to the file system at the end of the command execution.

3.3.2 Anti-tearing Mechanism

The anti-tearing mechanism is a mechanism to maintain the consistency in the file system against power cutoff or forceful removal of the card from the reader. The SCSOS implements anti-tearing mechanism through the cache. Data is first written to the cache instead of directly writing in to the file system during command execution. This data written is updated to the file system at the end of command execution. This mechanism ensures that if the command gets executed completely, only

then the file system is updated. Otherwise, during the command execution if the power cuts off or if the card is removed then the cache remains invalid. A valid cache is updated to the file system at each power on time.

3.4 Security Infrastructure of SCSOS

In the SCSOS, several resources such as files, data objects are protected against operations like read, write, update etc. The protection of resources is done using command level security and file level security. The command level security allows or disallows the command execution depending upon the permissions in the currently selected DF. While file level security is used for protecting files against unauthorized access to read, write and update files. The control attributes required for resource protection are specified in the FCP of the file.

SCSOS maintains the security status for keys and passwords that have been authenticated. The security status is maintained in the form of one bit per password and one bit per key at each level from MF to the currently selected DF. This security status is used for access control mechanism in the PIP.

3.4.1 Security Operations

The SCSOS supports encryption, decryption, checksum computation and verification operations. The algorithm reference and the key reference required for performing the security operation is taken from the current SE. Along with these basic security operations the SCSOS supports secure messaging for the command and response APDU protection. The session keys are required to be established for the secure messaging between the reader and the card which is done using mutual authentication algorithm. These session keys are used for performing encryption, decryption, checksum computation and verification operations during secure messaging.

3.4.2 Password and Key Repository

In SCSOS, the passwords and the keys are stored in an identical manner in the elementary files in the SCSOS. The elementary file with a short file identifier ‘1’ is used for the password repository while elementary file with short file identifier ‘2’ is used for the key repository. There can be multiple files

Algorithm Reference	**CRT template of the algorithm**	**Algorithm**
00(Default) or 01	CT	TDES based encryption and decryption
01	CCT	TDES based CBC residue
00 (Default) or 02	CCT	9797-1 Algorithm 3 for MAC using TDES
00 (Default) or 02	AT	TDES based challenge response
02	AT(AUTH)	ISO/IES 11770-2 Key Establishment mechanism 6 using 3DES
00 (Default) or 01	HT	SHA-1 as defiened in FIPS-140

Table 3.1: Algorithms in SCSOS

with same short file identifier but only internal record files are used for storing keys and passwords. These keys and password file can be under any DF or under MF termed as global keys and global passwords. Keys and the passwords under MF. Similarly keys and passwords in a DF are termed as local keys and local passwords. The SCSOS stores the keys and the passwords in the structural form as shown below.

- Structure of PIN/password repository

  1 byte PIN identifier,

  4 bits retry counter,

  4 bits maximum retry count,

  Variable length PIN.

- Structure of key repository

  1 byte key identifier,

  1 byte key type,

  Variable length key specific information,

  1 byte containing 0,

  Variable length key

Key type gives information about the operations on which key is permitted to be used as per SCOSTA-CL standard. The key specific information includes the usage counters for encryption, decryption and internal authentication which are monotonically decreasing counters. If the counter value is set to '0xFFFF' then the counter is considered as infinity.

3.4.3 Security Algorithms

The SCSOS has support for the algorithms as shown in table 3.1. The algorithms are identified using the algorithm reference.

The algorithms described in table 3.1 are used to carry out various security related operations and are also used in secure messaging.

Chapter 4

Implementation

The SCSOS is implemented in a generic manner. It has been ported to NXP P5 series processors including P5CD036, P5CD072, P5CD080 and as a simulator on Linux. The OS implementation is described here with reference to the functionality.

4.1 OS Initialization

The SCSOS is started through a bootstrapping procedure due to a power-on-reset. The run time system then leads it to the initialization for the OS, where several hardware specific parameters, OS specific parameters and communication mode etc. are initialized. The hardware specific initialization includes initialization of CPU and co-processors, security status, UART initialization [46] etc. OS specific initialization includes initialization related to the file system, current SE and synchronization of cache contents to the file system, if the cache was valid to implement anti-tearing mechanism. After hardware specific and OS specific initialization, communication mode specific initialization is done. The card may be started in contact mode or in contact-less mode and the initialization of communication parameters depends upon the mode in which it started.

4.1.1 OS Initialization for Contact Mode

In contact mode, the card is in physical contact with the reader and communication takes place through electrical connections between the reader and the card. When the card gets initialized during power

up, the processor is reset and it provides an initial sequence of bytes indicating the card capabilities. This sequence is known as Answer To Reset or ATR.

ATR carries several bytes such as interface bytes, historical bytes. Interface bytes encode the protocols supported by the card and the protocol specific parameters. The historical bytes encode other related optional information as specified in ISO 7816-3 standard. The structure of ATR along with detailed description of each byte is given in ISO 7816-3 standard.

In our implementation of SCSOS, T=0 and T=1 are the protocols supported in contact mode. The support for the protocols along with the global and protocol specific parameters for these protocols are encoded in appropriate ATR bytes. The ATR is prepared as per ISO 7816-3 standard and values supported by SCSOS. It is sent out to the reader using PDP and PIP modules of OS. The PIP module uses PDP for obtaining the part of the ATR which is specific to the processor. This part of the ATR string is pre-processed by the PIP using file system specific information such as the status of the MF and the checksum. The MF can be in creation, in initialization, in operational or in termination state as defined in ISO 7816-9 standard [41]. The ATR so prepared is transferred to the reader within a time limit as specified in ISO 7816-3 standard [39].

After ATR has been given out, the card may receive a protocol and parameter selection (PPS) to select one of the supported protocols. If the card receives PPS request then it is processed. The processing of PPS request involves protocol selection and adjustment of baud rates at the hardware level which is done by PDP. Finally, the PPS response with negotiated values is sent using output routines of PDP.

4.1.2 OS Initialization for Contact-less Mode

Contact-less mode uses radio frequency (RF) field for the communication from the reader to the card and vice versa. When contact-less card is brought in the RF field, it needs to be activated. The activation starts with contact-less reader polling for the availability of the card in the field by continuously sending the requests in the RF field. The card responds by sending response to this request. Further, the reader selects a card by sending another sequence of requests and obtaining their responses from the card known as anti-collision and select loop. After card selection, the card transmits an answer

to select byte string (ATS) indicating capabilities of the card. The ATS carries information about baud rate supported for the data transmission [39], maximum frame size that the card can receive etc. The PPS request can be received immediately after the ATS has been sent to select one of the communication setup.

The implementation of initialization in contact-less mode is done by PDP and PIP. PDP provides functionality to receive and transmit frames of all types mentioned in ISO 14443-3 [34]. ATS needs to be formed before transmitting it to the reader. The ATS is transmitted using transmit frame functionality of PDP. If the card receives the PPS then protocol specific parameters are negotiated. The PPS request has a parameter which represents baud rate value to be negotiated. The baud rate is adjusted at the hardware level based upon value to be negotiated and baud rates supported by SCSOS.

4.2 Transmission Protocols

SCSOS supports communication in contact as well as in contact-less mode. The data transmission in contact mode happens through either T=0 or T=1 protocol.

4.2.1 T=0 Protocol

The T=0 protocol is byte oriented data transfer protocol. The transmission data unit used for data transfer in this protocol is known as TPDU. The command TPDU consists of a header containing class byte, a command byte and three parameter bytes. The card first receives command TPDU and based upon these bytes it determines type of command. The types of command are discussed in chapter 2. For type III and IV commands, the data cycle starts after receive operation of TPDU to receive remaining bytes of command APDU. This data cycle receives remaining bytes indicated by third parameter of command TPDU.

The card processes the command and transmits the response in the form of response TPDU. The response TPDU consists of optional response data and two status bytes indicating the status of the command execution. In case of type I and III commands, the response TPDU consists of only status bytes while in case of type II commands response TPDU consists of response data along with the status bytes. For type IV command, the response is sent in two response TPDUs. In first response

TPDU, only status bytes are transmitted which may indicate availability of more bytes of response. If more bytes are available with the card then the card may receive second command TPDU with GET_RESPONSE [38] command to obtain the remaining bytes of response data. The response of this command is sent in second response TPDU which has response data and status bytes.

The implementation of this protocol can be broadly divided into implementation of receive command TPDU and send response TPDU operations. PIP code for these operations uses PDP input/output routines for receiving and sending TPDU. In receive/send TPDU operation is done one byte at a time. The TPDUs are received/sent based upon protocol rules.

4.2.2 T=1 Protocol

The T=1 protocol is asynchronous block half-duplex protocol based upon ISO 7816-3 standard. Blocks are used for data transfer from the reader to the card and vice versa. There are three types of blocks in this protocol– I-block, R-block and S-block. I-block is used to carry application specific data, R-block is used to convey positive and negative acknowledgments. S-blocks are used to exchange control information. Every block has prologue, epilogue and information fields. The information field of the block carries commands and responses.

The implementation of this protocol is done by Aditi Gupta as a part of course project [30].

4.2.3 Contact-less Protocol

Contact-less transmission protocol is a half duplex block oriented data transfer protocol. The blocks in the contact-less protocol are of similar types and serve the same purpose as described in T=1 protocol.

The commands and responses are transferred using I-blocks. Chaining is one feature which allows sending large data that does not fit in a single block. The data is transmitted by dividing it into multiple chunks which can be transmitted in a single block. The protocol rules to be followed by the card and the reader for block transfer and error handling are defined in ISO 14443-4 [37].

The implementation of contact-less transmission protocol is done in PIP and PDP. The PDP provides functionality to receive and send frames. The errors such as frame length error, receive error etc. are handled in send and receive frame operations.

Receive command APDU and send response APDU operations are the two operations based on which the implementation is divided. Command APDUs are received in chained I-blocks or in a single I-block. The protocol rules are followed upon reception of I-block, S-block and R-block.

The response of the command incorporates optional response data and mandatory status bytes. If the response fits in a single I-block then a single I-block is transmitted without chaining, otherwise multiple I-blocks are transmitted in a chain.

4.3 APDU Receive and Send Operation

The APDU receive operation is done in the form of TPDU in case of T=0 and T=1 protocols and in the form of blocks in contact-less protocol.

In case of T=0 protocol, TPDUs are sent between the reader and the card and they define the APDUs. The command TPDU can be directly mapped to command APDU except the third parameter byte (P3). The value of P3 byte in command TPDU can be either Lc or Le field depending upon the type of the command. The response APDU has optional response data along with the status bytes. The response TPDU is formed based upon response APDU and the type of command.

In case of T=1 protocol, blocks are sent and received as TPDUs for command and response data transmission. One or more I-blocks are used to transmit command APDU in the information field of the blocks. The response APDU (response data along with status bytes) is encapsulated in information field of the I-blocks and sent.

In case of contact-less protocol, the command and response APDUs are transmitted in blocks. Information field of I-block encapsulates command and response APDU as in the case of T=1 protocol.

If the command has secure messaging then SM is processed and the plain APDU is obtained.

4.4 Secure Messaging

The basic aim of secure messaging (SM) is to protect all or part of the command and response APDU and to make the communication between card and reader secure. The protection of command and response APDU is achieved by two basic functionalities – data confidentiality and data integrity. The

data confidentiality is achieved by data encryption while data integrity is achieved by checksum computation and verification over the command and response APDU. If the command indicates the presence of secure messaging, then command data field contains SM data objects, known as SM field.

There are two categories of SM data objects – basic SM data object and auxiliary SM data object. Basic data objects are defined in ISO 7816-4 for encapsulating plain data values, to provide confidentiality and to include checksum over SM field. The commands and their responses are encapsulated using the basic SM data objects. The auxiliary SM data objects are used to provide CRTs for performing confidentiality and integrity related operations on the SM data object.

The presence of SM field in command APDU is identified by CLA byte of the command APDU. It may incorporate confidentiality, integrity, both or none. When the SM does not incorporate integrity, then the command header is not included in the checksum calculation over command data field.

The SM data objects are processed for command APDU and response APDU as follows.

4.4.1 Command APDU Processing

The command APDU containing command data field, Le field, command header are encapsulated in secure messaging data objects (SM DOs). Possible SM DOs in which command data field can occur includes SM DOs with tag '80/81' for inclusion in plain data form. Otherwise, SM DO with tag '86/87' can also include command data field in an encrypted form. The Le field is included in the SM DO with '97' tag. Command header is included in SM DO with tag '89'. The constructed SM DOs can be used for encapsulating above mentioned DOs.

At the card side, if checksum data object (DO) is present then the checksum is computed over the SM data field and then it is verified with the value field of the checksum DO. If the checksum verification is successful, then SM DOs in the SM command data field are processed. The SM DOs are processed using generic security routines. The key and algorithm references required for the decryption and checksum verification are obtained from current SE. The command data field may contain CRT in SM DO form for these operations. After processing of SM DOs in the command data field, the command APDU should have appropriate fields depending upon type of command.

4.4.2 Response APDU Processing

The response APDU has optional response data and status bytes. Response data is sent in encrypted form if encryption key is available with required usage, otherwise it is sent by encapsulating in a plain DO. The status bytes of command are encoded in SM DO with '99' tag. The DO for checksum is included in the response if integrity key for SM operation is available in the current security environment (current SE) of the card. If integrity key is available then SM DOs with odd tags are used to encapsulate the response data.

4.5 Security Architecture

The security architecture of the OS is initially implemented by Ankit Jalote and Marghoob Mohiyuddin [42]. It is modified in the implementation part of this thesis in order to incorporate secure messaging support in OS.

The card maintains the security status for every file in the path from master file to currently selected file. Every DF has respective password and key file. At the maximum '32' passwords and keys are possible for each depth from master file to DF. The status for each key and password is kept with a bit per key and per password. If verify/external authenticate command succeeds then the corresponding bit for password/key is set to indicate key/password is authenticated. Along with the status for keys and passwords, the OS also stores the keys and algorithm references used in the SM session for command as well as response SM processing. When directory is changed the current security status is cleared on the path starting from the lowest common ancestor of the current and the previous directory till the previous directory.

The current security status maintained by the card is used by security check module. The security check module checks the file and command level security for each and every command passed to the card. The command execution permissions are checked first using the security status of the current DF and security attributes of a DF. If the security attributes of DF or EF has security condition for SM then the key reference and algorithm reference used in command and response processing must match with the key and algorithm reference with corresponding CRTs of current SE or SE specified

in security attributes of the file.

4.6 Security Environment

The SCSOS supports confidentiality template (CT), cryptographic checksum template (CCT), authentication template (AT) and hash template (HT). The templates are stored in security environment (SE) of the DF or in the FCP of the DF. The SE can be stored in security environment file. In SCSOS, SE file is an internal record file and SEs are stored as records which are accessed by record number.

At any point of time, the card has security environment set known as current security environment (current SE). In SCSOS, an array of structure is maintained for current SE. the structure stores key reference, algorithm reference, IV mode and key pointer pointing to key location in the file system for each of the operation such as encryption, decryption, CC computation, CC verification, SM encrypt, SM decrypt etc. Thus, whenever any of the security operation is to be performed current SE stored in the structure is accessed. The storing of the key, algorithm reference etc. corresponding to the security operation reduces the burden of extracting them from SE by scanning and checking SE for appropriate CRT usage qualifier. The current SE is managed through manage security environment command.

4.7 Manage Security Environment (MSE)

The SCSOS supports MSE SET and MSE RESTORE commands to mange current SE. The MSE RESTORE command is issued by the reader to set new security environment (SE) in the card. This new security environment acts as current SE of a card. The CRDOs of current SE are processed and structure of current SE is updated after each MSE RESTORE command.

The MSE SET command is used to update CRDOs in the CRT of current SE. Some CRDOs of current SE are changeable while some are non-changeable which is defined in SCOSTA-CL standard. This command is used to update changeable CRDOs of current SE. This command can be used to derive the key for confidentiality and integrity related purpose. The key usage for the derived key is specified in the command parameters. In the key derivation using this mechanism, the master key (key from which derived key is obtained by encrypting data) reference is obtained from appropriate CRT

based upon key usage and CRT usage qualifier. The data for encryption is taken from CRDO of same CRT or from command data field of MSE SET command. If the master key stored in key repository permits the key derivation then only key is derived.

MSE SET command is also used to set new value of IV and to set new IV update mode.

4.8 Key, IV and Algorithm Handling

An algorithm table is maintained in the OS which contains algorithm reference, control reference template (CRT), key size and block size. The key size and block size are the parameters on which algorithm operates at a time.

The direct usage keys (the key which can directly used without derivation) and master keys are stored in key repository of the file system. Keys are read from the key repository whenever they are required. The key structure used to store keys in the key repository is described in the previous chapter.

SCSOS maintains two derived keys, one for confidentiality and another for integrity. Both of these keys store key reference, key type and key value. The key reference of the derived key is same as key reference of master key. The derived key can be used only for the purpose it is derived for. This information is maintained along with the derived key in key type in order to permit key usage and do the error handling.

The initial value (IV) is kept for confidentiality and integrity separately. The IV update is done based upon IV mode. There are three types of IV modes– IV cipher block chaining (CBC) mode, IV null and IV increment mode. IV CBC mode updates IV with the last enciphered block of CBC encryption or CBC residue. IV increments IV value by 1 after each cryptographic operation. IV null initializes IV to zero.

The derived keys are obtained with MSE SET command or during the session key establishment for SM.

4.9 Session Key Establishment

OS maintains two session keys, one for the confidentiality and other for integrity for SM processing. When the keys are derived, the OS maintains the usage of the derived key as discussed in section 4.6.3. Session keys are derived either using authentication along with session key derivation algorithm or using MSE command.

4.9.1 Authentication along with Session Key Derivation

To use SM in the command and response APDU, session keys are derived. Session key can be derived using GET CHALLENGE and MUTUAL AUTHENTICATE command [38]. The authentication algorithm for session key establishment is described in ISO/IEC 11770-2 [36] which in turn uses 3DES.

The authentication algorithm establishes a session for SM communication by establishing two session keys. One key is used for confidentiality related operations such as encryption and decryption while other key is used for integrity related operations such as computation and verification of checksum. These keys are stored as confidentiality key and integrity key with appropriate key type. This algorithm also generates send sequence counter (SSC) which is used as an IV for integrity. This algorithm does mutual authentication i.e. authentication of the card and the reader because symmetric key is required in the processing of command. If the card and the reader does not have symmetric key then the command can not be executed successfully.

4.9.2 Session Key Derivation using Manage Security Environment (MSE) Command

The MSE SET command is used for the session key derivation for integrity and confidentiality related operations. The key is derived by encrypting the data with master key using TDES encryption algorithm. The data for key derivation is given in command data field or can be found in SE itself. The master key reference is taken from the CRT mentioned in the command parameters of MSE SET command with appropriate CRT usage qualifier to derive the key. The derived key is maintained by SCSOS in the array of bytes as described in section 4.6.3.

4.10 Generic Security Routines

The generic security routines are implemented in SCSOS to perform security related operations such as encryption, decryption, authentication, checksum computation and verification. These generic routines hide lower level details such as algorithm handling, key sanity checking etc. from programmer. Each of these routine performs some functionality required for the security operations. It also calls the algorithm handling routines. To add a new algorithm support in SCSOS, the routine to handle algorithm and its corresponding algorithm table entry needs to be added. This procedure makes it easier to add new algorithm support in OS.

In authentic generic routine, the algorithm reference and the key reference is taken from command parameters or current SE. This generic routine is called for internal, external and mutual authenticate commands. It also distinguishes external authentication and mutual authentication based upon the length of the command data field. This routine maintains security status of authenticated keys. The security status is cleared when authentication for a key fails.

For other generic routines, some of the common functionalities implemented for include padding of the buffer data, obtaining the index from algorithm table to get information about key length, block size and algorithm reference. This information is used to verify buffer length and key size with the buffer length and key length supported by the algorithm. For example, key size to be used must match with the length of key required by the algorithm. The generic routines calls security algorithm handling routines. It also updates IV based upon IV mode in the current SE for a particular operation. These generic security routines are used for the processing SM DOs, for performing security operations (PSO).

4.11 Operating System Efficiency

The smart card has limited memory and limited CPU execution capacity. Some of the efficiency techniques implemented in the SCSOS are discussed below.

The global variables are not initialized to zero at the time of declaration. This is done because global variables needs initialization of the memory segments in which they are kept before card gives

ATR. The timing requirement to give out an ATR might be missed due to initialization of memory segments.

The binary file reading with the read binary command is done through direct read operation of the file system instead of passing it through cache. The cache is initially synchronized with the file system prior to read operation. This process makes read binary operation faster.

The security environment in the card is processed in such a way that key, algorithm references etc. are extracted from security environment for each of the security operation. This reduces burden of extracting key, algorithm references etc. whenever performing any security operation.

The code size is kept to minimal possible. The size of code is 58KB.

4.12 Security Commands

Some of the security commands are implemented as a part of this thesis. The commands mentioned in this section are described in SCOSTA-CL standard [31]. Here, commands are discussed in implementation view point.

- PSO command

  The SCSOS allows applications to perform encryption, decryption, CC computation, CC verification and hash computation operations using perform security operation (PSO) command. The operations supported by PSO command are distinguished with the command parameters (P1 and P2). Implementation of these operations uses generic security routines. Keys and algorithm reference required for the above operations are taken from current SE.

- Authentication commands

  The SCSOS has two authentication algorithms, TDES based challenge response and ISO/IES 11770-2 key establishment algorithm. Whenever any authentication command is issued the generic authentication routine is called to execute the command.

- Verify

Verify command verifies the password stored in the card with the password passed in the command data field. If the password verification is successful then the status corresponding to the password is updated.

- Reset Retry Counter

  This command modifies retry counter of the PIN record. The record is updated if update operation is permitted on file which is checked by using current security status.

- Enable Verification Requirement

  The enable verification requirement command updates the valid bit of the password record. The update permission on the file are checked by using current security status, if the updates are allowed then command is valid bit is updated.

- Disable Verification Requirement

  Disable verification requirement command resets valid bit of password record to zero, rest of the command execution is same as disable verification requirement.

- Change Reference Data

  The command changes reference data such as passwords. The command data field may contain the old reference data along with new reference data. If old reference is present then it is verified with the reference data stored in the file. The security status is updated depending upon the verification status. If command contains only new reference data then the security status for the password must have been verified earlier prior to changing the reference data.

  Thus, the command updates the reference data with new reference data if the security status is satisfied and update permission on the file are allowed.

4.13 Security Algorithms

Some of the security algorithms are implemented to support secure messaging in the card. Following are the algorithms implemented.

IOS/IES 11770-2 key establishment algorithm 6 using TDES algorithm is described in ISO/IEC 11770-2:1997 standard [36] and also specified in machine readable travel document [33]. This algorithm uses ISO/IES 9797-1 algorithms 3 for MAC calculation and verification. The SSC is established in this algorithm. The implementation of this algorithm is discussed earlier in session key establishment for secure messaging.

ISO/IES 9797-1 algorithms 3 for MAC using TDES algorithm is described in ISO/IEC 11770-2:1997 [36]. It computes MAC over buffer of data. The MAC computation algorithm uses DES in hardware. The algorithm uses SSC established during the session key establishment algorithm. SSC is incremented before MAC computation starts.

SHA-1 as in FIPS-140 [26] algorithm computes hash of size 160 bit. The implementation of this algorithm is independent of endianity of the processor. It is optimized with least possible use of temporary variables and memory resources.

Chapter 5

Operating System Portability

The SCSOS smart card operating system code is structured in a way to make it easy to port it to different available processors. We have ported SCSOS on NXP P5 [47, 46] family of processors which include P5CD036, P5CD072 and P5CD080 processors [47, 46]. The SCSOS is also ported on P5CD144 [47] processor running EvalOS [49, 48] operating system for testing purposes. The EvalOS allows the code to be executed from the EEPROM and therefore the code may be compiled and run on actual processor prior to converting it to ROM. The design for portability requires certain disciplines in coding.

Different processors have different ATR requirements and therefore we implemented the ATR handling jointly in processor dependent and processor independent parts (PDP and PIP parts). The PIP sends the ATR whereas PDP provides the ATR which is first processed by PIP before sending. In our implementation, the PDP obtains the raw ATR string from a fixed area in the EEPROM which may be at different locations for different processors. PIP processes the raw ATR and sends actual ATR to the reader.

Every card has chip sequence number (CSN) associated with it, which is unique for a card. The CSN of a card is stored in a fixed area in a data object (DO). It can be obtained using the GET_DATA [37] command provided by SCSOS to get the value of data object (DO).

To port SCSOS to different processors, the processor dependent part (PDP) of the code needs modification. The size of PDP code has been kept as minimum as possible in order to make minimum

possible changes while porting SCSOS. We discuss the porting of SCSOS on P5 family of processors. This P5 family of processors support contact interface, contact-less interface, separate co-processors to implement Data Encryption Standard (DES) [28] and Advanced Encryption Standard (AES) [27]. These processors differ in size of EEPROM, EEPROM layout etc.

The processor dependent part (PDP) implementation can be broadly classified in following functional unit.

1. Initialization unit: The initialization unit initializes the control registers of the processor. These control registers sets the clock of CPU and co-processor. It also enables the signature calculation and the random number generation. The transmission related parameters are also set by the initialization unit.

2. Input/Output handling unit: This unit deals with receiving input from reader and transmitting output to reader. The send and receive operations are done one byte at a time. The bytes are read from and written to special function registers (SFR) [47] through which the input and output are handled.

3. EEPROM read and write handling unit: This unit deals with reading from and writing to EEPROM. The reading and writing is done through shadow page register which increases efficiency in reading and writing. The data is initially written to shadow page register and at the time of EEPROM programming data from shadow register is written to EEPROM. The EEPROM programming routine is explicitly invoked by the programmer to read or write content of shadow register to EEPROM.

4. Co-processor related handling unit: The crypto coprocessor such as DES and FameXE [47] are dedicated processors supported by P5 family to perform cryptographic operations using the symmetric and the asymmetric key cryptography. SCSOS supports symmetric key cryptography, therefore DES processor is used. The control registers are used to start the cryptographic operation. The result of cryptographic operations is read from the data registers of these processors.

5. Contact-less protocol handling unit: This unit deals with the send and receive operation of short

and standard frames in contact-less mode. The send and receive operations is handled at several baud rates supported by the reader and P5 family of processors. The buffer used for sending the frame is kept in faster memory segment to reduce the processing delay to read data bytes for transmission. This unit also handles anti-collision operation. The send and receive anti-collision frame handling is also done by this unit. The processor dependent data for contact-less protocol which include SAK, UID [34] etc. is located in the security area of a card. This area may be at different location for different processors. The area is used by this unit during anti-collision handling by this unit.

The above mentioned PDP functionalities are used by the PIP in the OS initialization and command processing. The minimum functionalities provided by the PDP keeps the code size minimal for PDP and make it easier to port SCSOS to different processor.

Chapter 6

Operating System Testing

Testing forms an integral part of any software development. The SCSOS is rigorously tested for error free command execution. Test cases were designed to test correctness of various modules of SCSOS such as ADPU receive and send modules, modules related to security architecture, modules related to secure messaging, etc. The testing process of SCSOS is evolutionary since more test cases are added in the existing test cases as new functionality and modules are added in the SCSOS. In this chapter testing, debugging tools used in testing and development of the SCSOS are described. Some of the test cases used to test the SCSOS are also mentioned.

6.1 Testing Tools

Tools used to test SCSOS include software simulators, hardware emulators and softmask cards. Software simulators work along with a hardware known as chip card to interface with standard smart card readers. The chip card and the software simulator together then act as a SCSOS smart card. Software simulators however runs at much slower speeds compared to actual processor. The timing related problems therefore, can not be caught with software simulators. Along with the timing problems, software simulators handle uninitialized variables differently compared to the actual processor. Therefore software simulators alone are not sufficient for testing. Hardware emulators operate at higher speed and memory is initialized to random values. This helps better in the testing process.

Hardware emulators are used to emulate the real processor hardware. The emulator therefore

needs to be configured for a particular processor before loading SCSOS. Typical hardware emulators work with the chip cards and provide the same interface as the contact and contact-less smart card. The emulator loaded with SCSOS software along with the chip card were used for testing and debugging purposes. The emulators can also be used for source level debugging.

Unlike emulators, the softmask cards are based on the actual processor but with different memory configurations. The software can be loaded in the EEPROM and executed from there. Softmask cards therefore essentially differ from real target with respect to the memory where the OS is loaded. In actual smart card, OS is written into the ROM whereas in softmask cards, the OS is loaded into EEPROM. The EEPROM is erasable making it possible to change the code and try it again. The purpose of loading the operating system on the softmask cards is to catch errors in the implementation which could not be caught otherwise. In our case, the softmask cards were pre-loaded with EvalOS operating system in ROM. Using EvalOS [49, 48], the customer operating system (COS) is loaded into the card in EEPROM and tested.

The SCSOS is tested by sending commands, obtaining their responses and comparing against expected value. The commands are sent to card as defined per the SCOSTA-CL standard. If the response received for the command matches with the expected response for the command then the test is considered successful.

We used several tools to test the SCSOS.

6.1.1 Smart Card Test

Smart card test is a small program which uses PCSC APIs [19] for sending commands and obtaining their responses. This tool helps developer to test the OS by sending set of commands manually and is suitable for small set of commands to be tried.

6.1.2 STTool

STTool is a tool developed by Venkat Rao Pedapati and Dr. Rajat Moona.[51] which tests the OS. It is an automated tool for testing purpose which accepts the test scripts and tests them on the smart card. This tool uses PCSC APIs [19] to transmit the command and to receive the response. STTool can send

APDU in the plain as well as in the SM form. The OS is rigorously tested using various test scripts developed for this purpose.

6.2 Test Cases

The OS is tested with several test cases. Some of the test cases include.

- Manage security environment command is tested for key derivation by giving master key reference which does not permit key derivation, correct key reference but making key encryption counter zero, etc.
- Authentication commands are tested by encrypting challenge with wrong key, making key counters for authentication commands zero, wrong challenge encryption, etc.
- Secure messaging is tested by encapsulating command data field in different SM DOs, giving wrong check sum, multiple command data fields with different SM DOs, giving wrong number of bytes in command header etc.
- Test to compute the read/write time with secure messaging with direct keys and derived key.
- Algorithm handling routines are tested with some of the available test cases.

The above mentioned test cases forms part of the test cases. In addition to these test cases more test cases are included to test file system implementation, security architecture etc.

The testing is carried out on contact and contact-less readers with different baud rates supported by SCSOS for contact-less protocol.

Chapter 7

Conclusion and Future Work

In this thesis, we described implementation of smart card operating system based on SCOSTA-CL standard. We have also discussed porting of operating system on several NXP processors belonging to the 8051 family. Some of the implementation features of the operating system include the following.

- The operating system has its own file structure and security architecture. The operating system provides security through password and key based authentication and maintains the status of the each key and passwords authenticated. The operating system grants read, write update etc. permissions if the required security status is satisfied.
- The operating system provides dual mode communication with the reader which includes contact as well as contact-less interface with the reader. The protocols for contact communication and contact-less communication are also discussed.
- Secure messaging support is implemented for secure transmission of command and response data. The secure transmission of commands and responses prevents eavesdropping during the communication. The secure messaging support for file and command level security has been added in the operating system.
- Generic security routines are implemented in the operating system to hide lower level details of key and algorithm handling for performing security operations.

- Efficient solution to handle keys, algorithm and current security environment has been implemented in the card. Some of the efficiency issues are also taken into account during implementation of OS.

- The operating system is rigorously tested for robustness and reliability. This operating system can be effectively used in variety of the applications like the passport application, the health care application, the student identity card application, the voting applications etc.

Future Work

The SCOSTA-CL standard supports symmetric key cryptography. The operating system based on SCOSTA-CL can be enhanced further to support applications which require asymmetric key cryptography. Asymmetric key cryptography can be used to perform key exchange of symmetric key, certificate verification, digital signature verification support etc. Asymmetric key cryptography along with symmetric key cryptography will provide comprehensive security infrastructure in the smart card.

The OS can have separate derived key repository supporting storage of more than two derived keys for confidentiality and integrity.

The file system in the operating system can have allocation strategies such as linked allocation or file allocation table (FAT) instead of contiguous allocation to avoid problem of external fragmentation.

Bibliography

[1] Advanced Card System (ACS) home page. *http://www.cryptoshop.com/index.php*.

[2] Basic card home page. *http://www.basiccard.com*.

[3] Europay International, Belgium home page. *http://www.europay.com/*.

[4] Forth programming language. *http://www.angelfire.com/in/zydenbos/WhatisForth.html*.

[5] Gemplus home page. *http://www.gemplus.com/*.

[6] Giesecke and Devrient GmbH, Germany home page. *http://www.gieseckedevrient.com/*.

[7] Java Card Forum, USA. *http://www.javacardforum.org/*.

[8] Mastercard home page. *http://www.mastercard.com/index.html/*.

[9] Microsoft home page. *http://www.microsoft.com/en/us/default.aspx*.

[10] Schlumberger Ltd., France home page. *http://www.slb.com/*.

[11] Siemens, Germany home page. *http://www.siemens.com/*.

[12] Simple Operating System for Smartcard Education (SOSSE). *http://www.mbsks.franken.de/sosse/html/index.html*.

[13] STARSIM cards. *http://www.gdburti.com.br/brochuras/starsim.pdf*.

[14] Visa International, USA home page. *http://www.visa.com/*.

[15] Windows Smart Card (WSC) home page. *http://www.microsoft.com/technet/security/guidance/identitymanageme*

[16] ZeitControl Cardsystems GmbH, Germany home page. *http://www.zeitcontrol.de*.

[17] ACOS. ACOS5 smart card reference version 0.5, November 09 2006.

[18] American association of motor vehicle administors. Smart card usage in Motor Vehicle Administration, January 1999.

[19] D. Corcoran. *MUSCLE PC/SC Lite API – Toolkit API Reference Documentation*. linuxnet, July 31 2000.

[20] EMV. Integrated Circuit Card Specification for Payment Systems, 2000.

[21] EMV. An introduction to emv, smart card industry standard, 2000.

[22] EN 1038. Identification Card Systems – Telecommunication Applications – Integrated Circuit Card Payphone, 1995.

[23] EN 1387. Machine Readable Cards – Health Care Applications – Cards:Genaral Characteristics, 1996.

[24] EN 1867. Machine Readable Cards – Health Care Applications – Numbering Systems and Registration Procedure for Issuer Identifiers, 1997.

[25] Ernst-G Giessmann. Specification version 1.0 of the security target tcos tachograph card. T-Systems International GmbH, 7 May 2004.

[26] FIPS 180-1. Secure Hash Standard (SHA-1), 1995.

[27] FIPS 197. Advanced Encryption Standard (AES), 26 November 2001.

[28] FIPS 46-3. Data Encryption Standard (DES), 1999.

[29] Giesecke and Devrient GmbH. STARCOS SPK 2.4 CHIP, FIPS 140-2 non-proprietary security policy version 1 validation, June 2002.

[30] A. Gupta. Implementation of T=1 protocol for Smart Card Operating System. CS697 Report, Dept. of CSE, IIT Kanpur, May 2007. *http://www.cse.iitk.ac.in/reports/*.

[31] IIT Kanpur and National Informatics Centre, Government of India. Specification for the Smart-Card Operating System with Contact-less Interface version 1.2, July 06 2007.

[32] IIT Kanpur and National Informatics Centre, Government of India. Specifications for the Smart-Card Operating System for Transport Applications (SCOSTA), March 15 2002.

[33] International Civil Aviation Organization. Machine Readable Travel Documents(MRTD) LDS for optimal capacity expansion technologies Revision 1.7, 18 May 2004.

[34] ISO 14443-3. Identification Cards – Contactless integrated circuit cards – Proximity Cards, Part 3: Initialization and Anticollision, February 2001.

[35] ISO 7813. Identification Cards – Financial Transaction Cards, 2001.

[36] ISO/IEC 11770-2:1997. Information technology – Security techniques – Key management – Part 2: Mechanisms using symmetric techniques, April 2002.

[37] ISO/IEC 14443-4. Identification cards – Contactless integrated circuit cards-Proximity cards, Part 3: Transmission protocol, February 2001.

[38] ISO/IEC 7816-3. Identification cards – Integrated circuit cards, Part 4: Organization security and commands for interchange, December 1997.

[39] ISO/IEC 7816-3. Information Technology-Identification cards – Integrated circuit cards with contacts, Part 3: Electronic signals and transmission protocols, January 2005.

[40] ISO/IEC 7816-8. Identification cards – Integrated circuit cards with contacts, Part 8: Security related interindustry commands, October 1999.

[41] ISO/IEC 7816-9. Identification cards – Integrated circuit cards with contacts, Part 9: Additional inter industry commands and security attributes, September 2000.

[42] A. Jalote and M. Moyuddin. Implementing the Security Module of a Smart Card Operating System. BTech Project, Dept. of CSE, IIT Kanpur, 2002. *http://www.cse.iitk.ac.in/research/btp2002/scosta.ps.gz.*

[43] H. G. Jurgen Dethloff. "Identifikanden/Identifikationsschalter". German Patent, DE 19 45 777 C2, February 1969.

[44] MPCOS. Mpcos-emv product overview version 1.0. by Gemplus, June 1999.

[45] H. Novinsky and H. U. Buchmuller. *Infineon Technologies AG Security and Chip card ICs SLE66 Processor version 1.1*. Infineon Technologies, 1.0 edition, 12 Sept 2005.

[46] NXP Semiconductors. *P5CD072 VOP/VOQ Secure Dual interface PKI Smart Card Controller product data sheet*, 3.0 edition, 06 March 2006.

[47] NXP Semiconductors. *P5Cx012/02x/040/073/080/144 family Secure dual interface and contact PKI smart card controller product data sheet*, 3.3 edition, 17 August 2007.

[48] NXP Semiconductors. *P5CD040/P5CD080/P5CD144 EvalOS V1.2.3(Evaluation Operating System)*, 2.1 edition, 21 December 2006.

[49] NXP Semiconductors. *How to use EvalOS v1.2*, 1.0 edition, October 2003.

[50] Olivier Dubuisson. ASN.1. Open Systems Solutions, 1999.

[51] V. R. Pedapati and R. Moona. *STTool User's and Developer's Manual*. CSE dept. IIT Kanpur, 2.0 edition, June 2007.

[52] PrEN 1545-4. Identification Card Systems – Surface Transport Applications – Part 4: Vehicle and Driver Licencing, 1997.

[53] W. Rankel and W. Effing. *Smart Card Handbook*. John Wiley and Sons, 2004.

[54] R. Shankesi. Developement of an Operating System for Smart Card. Master's thesis, Indian Institute of Technology Kanpur, Dept. of CSE, India, April 2002.

[55] Visa. Visa Smart Debit/Credit Certification Authority Service Description (VSDC) version 2.1, December 2005.