Specifications for the Smart-Card Operating System with Contact-less Interface

(SCOSTA-CL)

Version 1.2 July 06, 2007

National Informatics Centre
Ministry of Communication and Information Technology
Government of India

Indian Institute of Technology Kanpur

History

1.	March 15, 2007	First Release Version 1.0
2.	June 11, 2007	Public Release of Draft Version 1.0
3.	June 29, 2007	Public Release of Draft Version 1.01
4.	July 06, 2007	Public Release of Draft Version 1.2

Table of Contents

Specifications for the Smart-Card Operating System with Contact-less Interface (SCOSTA-CL)

Part I. Operating System Interface

1. Scope

The purpose of this document is to provide specifications of an Operating System that may be put into an integrated circuit that provides a contact, a contact-less, or a combination of the two interfaces. Such an integrated circuit may be embedded in applications that use contact-less operations such as in an e-passport or may be deployed in a contact-less ID application. It may also be used in a contact interface based applications. This specification provides details on the Operating System that shall be inter-operable and provide interface for the personalization and initialization. In particular the methods have been adopted to make it possible to deploy such ICC for ICAO-compliant reading of the e-Passport for border controls.

While doing so, care has been taken to ensure the following.

- Compliance to the relevant ISO standards
- Applicability of the OS to multiple applications.
- Compliance to ICAO standard for Machine Readable Travel Documents (MRTD) while ensuring Inter-operability in the personalization of the passport.

2. References

The SCOSTA-CL is compatible to the following ISO standards for the Smart Cards.

- ISO/IEC 7816-4:2005, Identification cards – Integrated circuit cards – Part 4: Organization, security and commands for interchange
- ISO/IEC 7816-5:2003, Identification cards – Integrated circuit cards – Part 5: Registration of application identifiers
- ISO/IEC 7816-6:2003, Identification cards – Integrated circuit cards – Part 6: Interindustry data elements for interchange
- ISO/IEC 7816-8:2004, Identification cards – Integrated circuit cards – Part 8: Commands for security operations.
- ISO/IEC 7816-9:2000, Identification cards – Integrated circuit cards – Part 9: Commands for card and file management.
- ISO/IEC 7816-15:2003, Identification cards – Integrated circuit cards – Part 15: Cryptographic Information Application.
- ISO/IEC 14443-3, Identification cards – Contactless integrated circuit cards – Proximity cards – Part 3: Initialization and anti-collision
- ISO/IEC 14443-4, Identification cards – Contactless integrated circuit

cards – Proximity cards – Part 4: Transmission Protocol

The following references on the machine readable travel documents are also relevant.

- ICAO LDS 1.7—2004-05-18: Machine Readable Travel Documents: Development of a Logical Data Structure – LDS for optional capacity expansion technologies. Revision 1.7
- ICAO Annex I—Version 4.0—2004-05-05: Use of Contactless Integrated Circuits in Machine Readable Travel Documents
- ICAO Supplement 9303—Version 2005-4 V3.0—2005-06-12: *Machine Readable Travel Documents: Supplement 9303.*
- ICAO Technical Report—Version 1.1—2004-10-01: Machine Readable Travel Documents: PKI for Machine Readable Travel Documents offering ICC Read-only Access.

3. Introduction

This document specifies the operating system interface for the multi-purpose Smart Cards intended to be used in the context of the applications in India. It covers the interface details of the smart card operating system. However the implementation of the operating system, processor for the smart card and other such details are not covered in this document and are out of its scope. The physical communication interface details between the ICC and IFD are out of scope of this document. In general SCOSTA-CL compliant OS based ICC may work with an IFD with T=0, T=1, or ISO14443-type A, or type B, or similar protocols. Certain examples included here are for illustrative purposes only for better understanding and do not suggest any particular implementation on the applications.

The SCOSTA-CL describes the minimum support for the application using the Smart Cards. Operating systems with extra features not listed here will be SCOSTA-CL compliant only if they support all the features listed in this document and the extra features are non-conflicting with the functionality as provided in this document. A SCOSTA-CL compliant operating system will also be compliant to the ISO7816-4, -6, -8, -9. The physical communication interface may be compliant to ISO7816-3, ISO14443-3 and -4 standards.

4. Symbols and Terminology

The terminology used in this document is the same as the one used in the ISO/IEC and ICAO documents and the usual references are marked in this document to the terms and commands in the appropriate ISO/IEC and ICAO documents. In addition, the following is used.

'..' A number in hexadecimal representation (*e.g.* '30' is same as 48)

AM Access mode byte

AM_DO Access Mode data Object

APDU Application Protocol Data Unit. From an application's perspective, this

is the frame that is used for communication between an application running on an IFD and execution layer of the OS on ICC. In reality, an APDU may be translated.

AT Authentication template

AT (AUTH) AT template with usage byte to indicate internal and/or external authentication.

AT (USER) AT template with usage byte to indicate user-knowledge based authentication.

CC Cryptographic Checksum

CCT Cryptographic checksum template

CLA Class byte for the APDU

CT Confidentiality template

DATA Data for the command APDU

DF Dedicated file

DST Digital signature template

EF Elementary file

EF*i* Represents an EF with SFID of *i*. $30 \geq i \geq 1$

FCP File control parameters

HT Hash template

ICC Integrated Circuit Card (*aka* smart card)

IFD Interface Device (*aka* smart card reader). This term is used in a generic manner as contact based IFD or a contact-less PCD.

INS Instruction code for the APDU

KAT Key agreement template

KD Key for Derivation use

Lc Length of DATA in APDU

Le Maximum length of RESPONSE expected in APDU

MAC Message Authentication code

Message Digest A one-way function to calculate MIC from a message

MF Master File

MIC Message integrity code (*aka* MAC)

P1 First parameter byte for the APDU

P2 Second parameter byte for the APDU

PCD Proximity Coupling Device used for interactions with contact-less ICC.

RESPONSE Response data from the ICC for command APDU

RFU Reserve for Future Use

SC Security condition byte

SC_DO Security condition data object

SE Security environment

SFID Short file ID

SHA1 Message Digest Algorithm (Secure Hash Algorithm variant 1)

SW1 First byte of the status word in the response ADPU

SW2 Second byte of the status word in the response APDU

tAPDU Transport layer APDU. An APDU may be subjected to secure messaging. In absence of secure messaging, the tAPDU is identical to APDU. Otherwise, this represents a secure APDU.

tCLA Class byte for the tAPDU

tDATA Data for the command tAPDU

tINS Instruction code for the tAPDU

tLc Length of tDATA in tAPDU

tLe Maximum length of tRESPONSE expected in tAPDU

tP1 First parameter byte for the tAPDU

tP2 Second parameter byte for the tAPDU

TPDU Transport Protocol Data Unit. These are application level frames that represent the communication between the IFD and ICC. TPDU are derived from tAPDU as per the transport protocol.

tRESPONSE Response data from the ICC for command tAPDU

tSW1 First byte of the status word in the response tADPU

tSW2 Second byte of the status word in the response tAPDU

5. Application Protocol Data Unit (APDU)

For the sake of this document, we shall define the following two kinds of APDU.

APDU for interface to the transport layer (T=0, T=1, or contact-less type A or type B as defined in ISO/IEC 7816-3 and ISO/IEC 14443-3/-4). We shall call this as transport APDU or tAPDU in this document. The application level APDU (or just the APDU as referred to in this document) will represent the interface between the application and the OS.

tAPDU will have a command header, an optional data field and an option expected length parameter. The command tAPDU will contain the following.

- tCLA: CLA byte for the transport APDU
- tINS: INS byte in the transport APDU
- tP1: P1 byte in the transport APDU.
- tP2: P2 byte in the transport APDU

- tLc: Lc byte related to the transport APDU. Will be present only when the tAPDU needs command bytes.
- tDATA: Transport layer data bytes (of size 1..tLc).
- tLe: Field in the transport APDU to indicate number of bytes expected in the response tAPDU (transport layer response APDU).

Related fields in the application level command APDU are CLA, INS, P1, P2, Lc, DATA (1..Lc) and Le.

The response tAPDU in the transport layer will contain the following fields.

- tRESPONSE: Transport layer response (of size 1..tLe or less)
- tSW1, tSW2: transport layer status bytes.

Related fields in the application level response APDU are the following.

RESPONSE (1..Le or less), SW1, SW2.

tAPDU and APDU will have relationship based on certain parameters. If the Secure Messaging is not used, the tAPDU shall be identical to the APDU. If the secure messaging is used, tAPDU shall encapsulate the confidentiality and integrity of the application APDU as is indicated by tCLA.

From the application view point, the maximum size of the Lc and tLc will never exceed 255. The maximum size of the Le and tLe will never exceed 256. All such fields shall be coded in one byte only. The ICC therefore must support short-field Lc and Le and optionally may support extended-field Lc and Le.

6. Transport Protocol Data Unit (TPDU)

Transport Protocol Data Unit (TPDU) shall be used to carry the tAPDU from the IFD to the ICC for command and from ICC to the IFD for the response. Conversion from tAPDU to the APDU shall be carried out completely by the ICC (for the commands) and IFD (for the response) locally. The conversion from APDU to the TPDU shall be based on the transport protocol. For T=0 and T=1 protocols, such conversions are defined in the ISO/IEC 7816-3 and for contact-less protocols, such conversions are defined in the ISO/IEC 14443-3/-4 and ISO7816-3.

7. Basic Data Structures

SCOSTA-CL will support the following two categories of the files.

- Dedicated Files (DF)
- Elementary Files (EF)

The structure of the file system will be that of a tree with depth restrictions, if any, not less than 4 (including the MF and EF). Thus it will be possible to have at least one MF, at least one DF under the MF, at least one DF under this DF and one or more EFs under the last DF. At each node of the tree, there can be several children nodes, either of type DF, or of type EF. The root of the tree is the Master File (of type DF). At any node it should be possible to create

any number of children nodes (both DF and EF put together). The restrictions on the tree structure may be put only because of the limited size of the memory. Under this condition, a CREATE FILE command may fail with an error.

The size of the files will be static and will depend upon the values provided implicitly or explicitly through the File Control Parameters (FCP) declared at the time of CREATE FILE command.

Initially there will be no MF in a blank SCOSTA-CL-compliant ICC. At this time, the ICC shall execute no commands except the following.

> CLA=00, INS = 'E0', P1 = 00, P2 = 00. Data field for the command shall indicate the FCP related to the MF.

All commands except CREATE FILE (MF) shall return identical error conditions and fail to execute till the MF is created. MF once created, can not be deleted.

The ICC will support data layout for multiple applications each of which may be identified by a DF. The files under the MF will be global for all applications. The files specific to an application shall be stored under a DF node in the file system tree. An application may have sub-applications each represented by an individual DF under the parent DF. In such a case, the application may organize the files as global within the application and local for each sub-application.

7.1 Elementary Files

The elementary files (EF) will be used to keep the data pertaining to an application. An EF can be one of the following types.

- Transparent EF (as defined in ISO/IEC 7816-4)
- Linear EF with fixed size records (as defined in ISO/IEC 7816-4)
- Linear EF with variable size records (as defined in ISO/IEC 7816-4)
- Cyclic EF with fixed size records (as defined in ISO/IEC 7816-4)

The OS will use some EFs internally as well. A few pre-determined and fixed short EF identifiers may be used to identify such EFs. There is however no restriction on the 16-bit identifiers for the same. In particular the OS uses short EF identifiers 1 and 2 internally. Section 10 of this document describes their use. A file whose contents may be used by the OS internally will be declared as INTERNAL EF.

7.2 File Referencing Methods

Each file (an EF or a DF) will have one 16-bit file identifier. The following restrictions will apply on the 16-bit file identifiers.

- '0000' and 'FFFF' shall be reserved and shall not be used as identifiers.
- '3F00' will be reserved for the MF and will not be used for any other file.

- ‘3FFF’ will be reserved to indicate the current DF and will not be used for any file.
- ‘2F00’ and ‘2F01’ will not be used for any general file under the MF where they are reserved for specific use as defined in ISO7816-4.
- 16-bit File Identifiers will be unique among the identifier for EFs and DFs under a single DF.

The EFs can also be referenced using another short EF identifier (5-bits) given at the time of CREATE FILE command. The following restrictions will apply on 5-bit short file identifiers.

- Short EF identifiers shall be used only for the files under the currently selected DF.
- Short EF identifier of ‘00’ and ‘1F’ will be reserved and will not be used for any EF. In particular, short EF identifier of 0, when used, will refer to the currently selected EF. Short EF identifier of ‘1E’ under the MF will also be a reserved one as per ISO/IEC 7816-4 and shall not be used for any other general file by the applications.
- Short EF identifier may not be unique among the EFs under a DF.
- Short EF identifiers will not be assigned to a file when tag ‘88’ with length 0 is used in the FCP for the file during CREATE FILE command.
- Short EF identifier will be assigned to the file when tag ‘88’ with length 1 is used in the FCP and the value field indicates a number between 1 and 30. If the value field is a number other than between 1 and 30 (both inclusive), an error shall be returned by the CREATE FILE command.
- Short EF identifier will be assigned to an EF, when tag ‘88’ is not used in the FCP, as a value given in the least significant five bits of the 16-bit file identifier provided it falls between 1 and 30 (both inclusive). In case, this value is 0 or 31, the short EF identifier shall not be assigned.
- While selecting a file using short EF identifier, the results will be unpredictable if two or more files exist with the same short EF identifier.
- Under any DF, there can be a maximum of one EF which is of type internal and has an SFID=‘01’. Same restrictions will also apply for an internal EF with SFID=‘02’.

Each DF in the file system may also have a DF name that shall be unique among all DFs in the file system. This DF name can be specified in the FCP during the CREATE FILE command. It will be possible to refer to a DF with its name irrespective to its location in the file system when the name was specified during the CREATE FILE command. When the DF name is not specified, it will not be possible to refer to that DF by its name.

Files can be referenced using the following four methods as described in the ISO/IEC 7816-4 section 5.3.1.

- Referencing by the file identifier. The values ‘3F00’, ‘3FFF’ and ‘FFFF’ are used for special purposes as given in ISO/IEC 7816-4 standard. In

addition to this, the file IDs ‘2F00’ and ‘2F01’ under the MF will also be reserved.

- Referencing by path. The path starting with ‘3F00’ shall refer to the absolute paths (i.e. starting from the MF) while the paths starting from any other file id will refer to the relative paths (i.e. starting from the current DF). Paths starting from ‘3FFF’ will also refer to the paths starting from the current DF. ‘3F00’ and ‘3FFF’ can occur only in the beginning of a path. For example, a path such as ‘3F003F00’ shall be considered as a wrong path.
- Referencing by short EF identifier. Valid SFID will be a value in the range 1 to 30 (i.e. ‘01’ to ‘1E’) and will be coded appropriately in the commands. A SFID of 0 will refer to the currently selected EF. There may be multiple files with the same SFID.
- Referencing by DF name. DF name referencing will be possible only for the DFs which were created with the name given in the FCP. The file name shall be unique among all DFs in the ICC. In case, two DFs are given the same name, the behavior of the OS is not specified.

7.3 Data Referencing Methods

A SCOSTA-CL compliant operating system shall implement at least the following data referencing methods as defined in ISO/IEC 7816-4 section 5.3.2.

- Record referencing
    - Referencing by one byte record number
    - Referencing by one byte record identifier

  It is implied in this document that the maximum size of a record shall not exceed 255 bytes.
- Data Unit referencing

  The data unit referencing shall be available for the transparent EF. The operating system may implement data unit referencing mechanism for the record structure files as well primarily intended to debug the applications. However in such a case, the data returned may include the structural information (meta-data) as well. SCOSTA-CL compliant applications will make no assumptions about the structure of the meta-data.

  The ICC may support variable data unit size (among these support for 8-bits wide data units is mandatory for all SCOSTA-CL compliant ICC) as defined in the table number 87 of ISO/IEC 7816-4.

7.4 File Control Information.

SCOSTA-CL compliant OS will support at least the FCP templates to be returned to the application during the SELECT FILE command execution. In case SELECT FILE command demands other templates, meaningful default

information should be supplied if the other templates are not supported.

8. Security Architecture

In SCOSTA-CL, various resources may be protected against various operations. These resources include files, DOs and commands. While checking for the security applicability, first the command level security shall be applied and only if successful, the individual resource level security will be applicable. For example, if a file is permitted to have a read access but the execution of READ BINARY command is subjected to certain security condition, the file can only be read if the READ BINARY security condition is met. SCOSTA-CL supports the secure messaging and therefore it will be possible for a resource to be utilized only under SM commands.

SCOSTA-CL compliant OS will support the security architecture given in sections 8.1 to 8.6 and as defined in ISO/IEC 7816-4, ISO/IEC 7816-8 and ISO/IEC 7816-9.

8.1 Security Status

The SCOSTA-CL compliant OS will support the following three kinds of security status.

- Global security status (related to the MF authentication processes)
- File-specific security status (related to the DF authentication processes)
- Command specific security status (related to the command authentication processes which are specific to the currently selected DF)

8.2 Security Attributes

It will be possible to attach security attributes to the files using the FCP (as given in ISO/IEC 7816-4). Both compact and expanded forms of defining security attributes must be supported.

SCOSTA-CL compliant OS shall support the access mode bytes (AM bytes) as defined in ISO/IEC 7816-4 for EF and DF. Other kinds of AM bytes (for example AM bytes for DOs and tables etc.) are optional for SCOSTA-CL compliant OS. No proprietary coding shall be used in AM/SC bytes.

There may be several SC bytes for each AM byte when the security specifications are specified in the compact form (tag ‘8C’ in the FCP). The number of SC bytes depends upon the number of bits set to 1 in the corresponding AM byte. Under tag ‘8C’ there may be several groups of AM and associated SC bytes. In case, an operation is covered by more than one such group, an OR condition shall prevail, i.e. any of the conditions may be satisfied. In a similar manner, when the security specifications are specified using expanded form (tag ‘AB’), there may be several groups of AM_DO and SC_DOs. In such cases when an operation is covered by several of these groups, an OR condition shall prevail.

In a group if the AM_DO provides an AM byte, the number of SC_DOs shall be equal to the number of bits set to 1 in the AM byte. In all other cases, an AND condition shall prevail if multiple SC_DOs are specified.

Security of the command can only be specified using the expanded form. Security of the file operations can be specified using the compact form and when it refers to a particular SE, the SE shall be looked for the CT and CCT tag(s) for the SM related security and AT tag(s) for the authentication (entity or user authentication). The usage qualifier bytes shall have to be consulted for the considered CRT.

Security of command shall be based on the currently selected DF only. That is only the security specifications for the currently selected DF and the resource will be looked at for the security.

If a DF or EF is in the terminated state, the only operations possible on it are selection and deletion. If a DF or EF is in the deactivated state, the only operations possible on it are selection, activation, termination, and deletion. Further, no commands are permitted if a DF in the terminated or deactivated state is the current DF. Any command that specifies a file (EF or DF) by path shall fail if a terminated or deactivated DF is one of the components but not the final component in the path.

8.3 Security Environments

The security environments (SEs) shall be stored either in an EF or in the FCP of the DF with a tag ‘7B’. In case the SEs are stored in an EF, the 16-bit file id of the EF shall be notified with tag ‘8D’ in the FCP of the corresponding DF. Such an EF can only be under the DF for which the SEs are being defined. When a DF is selected, the SE#1 for that DF will be selected as current SE automatically. In case, there is no definition of the SE#1, the current SE in force will be all empty. When the DF is created first and the corresponding SE file is not yet created the current SE will be all empty (since the created DF becomes the current DF). Upon selection of such a DF, the current SE will also be all empty. The current SE will not be modified when the SE file is created and written into. It shall only be modified when the DF is selected again.

The SEs are also selected using MANAGE SECURITY ENVIRONMENT (MSE) command. When a DF is selected or when the current SE is changed using MSE:RESTORE command, the definition of the previously selected SE will not be in force. In this case, the IV, IV update mode, derived keys, challenge, etc. shall be derived from the values provided in the new SE definition, or reverted to the default in case they are unspecified in the new SE definition. A SCOSTA-CL compliant OS shall support at least the following CRTs in the definition of the SEs.

- Confidentiality Template (CT). CT shall be identified by tag ‘B8’. CT will be subjected to the usage qualifier byte. If not specified, a default value of ‘C0’ shall be assumed (i.e. ENCIPHERMENT and DECIPHERMENT).
- Cryptographic Checksum Template (CCT). CCT shall be identified by

tag ‘B4’. CCT shall be subjected to the usage qualifier byte. If not specified, a default value of ‘C0’ shall be assumed (i.e. COMPUTATION and VERIFICATION of cryptographic checksum)

- Authentication Template (AT). AT shall be identified by tag ‘A4’. AT can be defined for the cryptographic authentication and/or user knowledge (PIN/Password) based authentication based on the usage qualifier. If the usage qualifier is not specified in the CRT, it shall default to ‘C0’ (i.e. EXTERNAL and INTERNAL CRYPTOGRAPHIC AUTHENTICATION)
- Hash Template (HT). HT shall be identified by tag ‘AA’.

A SCOSTA-CL compliant OS may support additional optional CRTs (for example for DST etc.)

8.4 Security Algorithms

SCSOTA-CL compliant OS shall support at least the following algorithms for message digest, confidentiality, integrity and authentication.

Table 1: Security Algorithms in SCOSTA-CL

Algorithm Reference Byte Value	CRT Template to which applicable	Algorithm
00 (Default) and 01	CT	3DES (Enc and Dec)
01	CCT	3DES based CBC Residue (CC Computation and Verification)
00 (Default) and 02	CCT	ISO/IEC 9797-1 Algorithm 3 for MAC using 3DES
00 (Default) and 01	AT (AUTH)	3DES based challenge response
02	AT (AUTH)	ISO/IEC 11770-2 Key Establishment Mechanism 6 using 3DES
00 (default) and 01	HT	SHA-1 as defined in FIPS-140

Details about these algorithms are given in section Part II.4.

8.5 Security Mechanisms

The SCOSTA-CL compliant OS shall support at least the following security mechanisms as described in section 5.2.3 of ISO/IEC 7816-4.

- Entity authentication with password (VERIFY command)
- Entity authentication with key (EXTERNAL AUTHENTICATE, INTERNAL AUTHENTICATE and MUTUAL AUTHENTICATE commands). These mechanisms will be subjected to the algorithms as defined in section 8.4 and may also include the facility to be able to establish a session key.
- Data authentication (computation/verification of cryptographic

checksum). These mechanisms will be subjected to the algorithms as defined in section 8.4.

- Data encipherment and decipherment. These mechanisms shall be subjected to the algorithms as defined in section 8.4.
- Computation of Message Digest (Hash) code for the data. The mechanism for the computation of hash shall be subjected to the algorithms as defined in section 8.4.
- Secure Messaging with confidentiality and Integrity requirements. Secure Messaging for the SCOSTA-CL compliant OS is explained later in this document. These mechanisms shall be subjected to confidentiality and integrity algorithms as defined in section 8.4.

8.6 Access Rule References

The access rule (in expanded or in compact format) for a file can be stored in the file's FCP. These rules shall specify under what security conditions certain operations on that file are permitted. The access rules for the DOs can be stored in the FCI of the corresponding DF. The access rule referencing mechanisms shall be according to the ISO/IEC 7816-9.

9. APDU Message Structure

SCOSTA-CL compliant OS will support at least the normal 1-byte fields for coding Le and Lc (tLe and tLc for tAPDU) fields in the command APDU/command tAPDU (as defined in ISO/IEC 7816-4). An APDU can be communicated with or without secure messaging. In case of secure messaging the corresponding tAPDU received shall contain one or more security mechanisms applicable to the command and the ICC shall send the response tAPDU with applicable secure messaging as defined by the received command for which the response is generated.

9.1 Coding Conventions for Command Headers

The command header has four bytes (CLA, INS, P1 and P2 for the APDU and tCLA, tINS, tP1 and tP2 for the tAPDU).

9.1.1 Class byte (CLA and tCLA)

The class byte coding as given in table 2 will be applicable to the SCOSTA-CL compliant OS.

Table 2: Coding for the class byte in SCOSTA-CL compliant OS

Class Byte								Meaning
b8	b7	b6	b5	b4	b3	b2	b1	
0	0	0	0	0	0	0	0	Commands in plain with no secure messaging
0	0	0	0	1	1	0	0	SM with authenticated command header
0	0	0	0	1	0	0	0	SM without authenticated command header

Others	RFU for SCOSTA-CL

In particular tCLA will have a value of ‘00’ or ‘08’ or ‘0C’. The CLA will have support only the 00 value.

9.1.2 Instruction byte (INS, tINS)

Instruction bytes will be coded as defined in the ISO/IEC 7816-4, -8 and -9 in the appropriate places. In this version of SCOSTA-CL only even INS bytes are to be used. Provision of odd INS byte codes shall not result in non-compliance to this version of SCOSTA-CL. However it may lead to non-compliance with future versions. The instructions that shall be implemented as mandatory instructions in SCOSTA-CL compliant OS are described in this document.

In case the secure messaging data objects contain the command header (under tag ‘89’), the INS byte shall be taken from that data object and not from tINS. Otherwise the INS byte shall be same as the tINS. In the former case, the command may be protected from the eavesdropper wherein the transport layer instruction byte (tINS) has no resemblance to the command being executed.

9.1.3 Parameters to the instructions (P1, P2, tP1 and tP2)

P1 and P2 bytes will be coded as described for the individual instructions in ISO/IEC 7816-4, -8 and -9 standards. The further interpretation of these bytes and restrictions if any are given in this document at appropriate places. In case the SM data objects contain the command header (under tag ‘89’), the tP1 and tP2 bytes shall be ignored and the P1 and P2 bytes shall be taken from the command header in the SM data objects. Otherwise the P1 and P2 shall be identical to tP1 and tP2.

9.1.4 Data field (DATA, tDATA)

The data fields of the command APDU and command tAPDU shall be coded as per the transport protocol of ISO/IEC 7816 and ISO/IEC 14443 standards. When there is secure messaging in force, tDATA field shall be a collection of SM data objects which shall be made from one or more of the following fields of the APDU.

- Command Header (CLA, INS, P1 and P2)
- DATA
- Le
- Cryptographic checksum of one or more fields.

If secure messaging is not used, the tDATA field shall be identical to DATA field.

9.1.5 Response data field (RESPONSE, tRESPONSE)

The data fields in the response APDU and response tAPDU shall be coded as

per the transport protocol of the ISO/IEC 7816 and ISO/IEC14443 standards. In case secure messaging is used the tRESPONSE shall be a collection of SM DOs and will be derived from the following fields of the response APDU.

- RESPONSE
- SW1 and SW2
- Cryptographic checksum of one or more fields.

In case, secure messaging is not used, the tRESPONSE field shall be identical to RESPONSE field.

9.1.6 Status bytes (SW1, SW2, tSW1 and tSW2)

These bytes are coded as described in the ISO/IEC 7816-4, -8 and -9 standards and suitably interpreted in this document at appropriate places. If secure messaging is used, the tSW1 and tSW2 shall carry only the secure messaging related status (i.e. '6987': SM Data Item missing; and '6988': Wrong SM Data). A SM data object in the response of tAPDU will provide the SW1 and SW2 (processing status of the command). If SM is not used, tSW1 and tSW2 shall provide the SW1 and SW2.

10. Password and Key Repositories

In SCOSTA-CL, PIN and passwords are stored in identical manner and therefore are indistinguishable. In this document, terms "password" and "PIN" are used in a generic manner to indicate PIN as well as Password and will be used for User-knowledge based authentication. Passwords and keys are stored in elementary files in SCOSTA-CL compliant OS. The following rules shall apply.

- The password are stored in an Elementary file EF1 (file with a short file id of 1) at various levels of directories.
- A valid password repository will be an INTERNAL record file (variable or fixed size).
- There can only be one password repository under any DF. (There can be multiple files with short file id of 1 but only one of them can be an internal record file).
- The global passwords are stored in password repository (EF1) under the MF.
- When the current directory is MF, global and local passwords will mean the same.
- There can be up to a maximum of 31 records in any password repository (i.e. EF1). These can store passwords whose qualifier can have a value between 1 and 31 (both inclusive).
- The keys are stored in a key repository in an Elementary file EF2, a file with a short file id of 2 at various levels of directories.
- A valid key repository will be a record file (variable or fixed size) and

will be an INTERNAL EF.

- There can only be one key repository under any DF. There can be multiple files with short file id of 2 but only one of them can be an internal record file.
- The global keys are stored in EF2 under MF. When the current DF is the MF, the local and global keys will be the same.
- There can be a maximum of 31 keys in a key repository. These can store keys numbered 1 to 31 (both inclusive).
- The OS and/or applications may choose any 16-bit file ids for password and key repositories.

As passwords are verified, the SCOSTA-CL compliant OS shall maintain the status of those passwords (verified/not-verified) at each level from the MF to the currently selected DF. Similarly as keys are authenticated, the SCOSTA-CL compliant OS shall maintain the status of those keys (authenticated/non-authenticated) at each level from the MF to the currently selected DF. The statuses of the keys and passwords for a DF are maintained when currently selected DF is modified (possibly due to the SELECT FILE command) if the DF lies in the common part of the paths from the MF to the old DF and from the MF to the new DF. The status of keys and passwords for all other DFs are treated as “not-verified” and “non-authenticated”.

A password corresponding to password reference specified in the password related commands such as a VERIFY command, or in an SE under the AT template with User-knowledge based key reference, may not be found. Similarly, a key reference specified in a key based command such as in INTERNAL/EXTERNAL/MUTUAL AUTHENTICATE command, or in an SE under any of the CRT that refers to the cryptographic keys may not be found. This condition may happen due to one of the following reasons.

1. The corresponding password or key repository does not exist.

2. The repository exists but no record is found corresponding to the referred key or password.

In such a case, the corresponding commands shall return a failure condition (‘6A88’, for “reference data not found”) whenever applicable.

The changes in the password/key may or may not be permitted depending upon the security attributes for that file.

10.1 Password Repository

The password repository will be a variable/fixed record file (up to a maximum of 31 records) with the following data items (in the same order).

Pin identifier	1 Byte
Retry counter	4 Bits (see below)
Max retry count	4 Bits (see below)
Pin	Variable length

The Pin identifier will be coded as follows.

b8	b7	b6	b5	b4	b3	b2	b1
V	RFU				Ref Data Number		

V bit represents if the corresponding entry is valid (1) or not (0). The password related commands such as VERIFY command for the invalid data will return a failure ('6984', "Reference data invalidated") in the status bytes. For the purpose of security attributes of a file operation, an invalid password or a non-existent password is considered as "VERIFIED".

V bit is manipulated using the ENABLE VERIFICATION REQUIREMENT and DISABLE VERIFICATION REQUIREMENT commands.

Records in the EF1 will have one byte containing the Retry counter and Max retry count. The bits b8:b4 of this byte will provide the Retry counter while bits b4:b1 will provide the Max retry count. Bits b8 and b4 will be the MSB of their respective fields. A value 'F' for the Max Retry count with non-zero Retry counter shall mean that there is no limit on the retries.

If Max retry count is a value other than 'F', then upon a successful verification, Retry counter for that reference data is set to the Max retry count. Also upon unsuccessful verification, Retry counter decrements by 1 and when it reaches a value of 0, subsequent VERIFY commands return a failure without any verification.

If Max retry count is 'F' (but the Retry counter is non-zero) then no limit will be set on the number of retries. In such a case, the Retry counter will not be changed after a successful or unsuccessful verification.

If the Retry counter is zero, then VERIFY command returns a failure without any verification irrespective of the value of the Max Retry Count.

10.2 Key Repository

The key repository is an internal EF containing variable/fixed records (up to a maximum of 31 records) with the following data items (in the same order) in each record.

Key identifier	1 Byte
Key Type	1 Byte
Key Specific Information	Variable length
RFU Byte	1 Byte
Key	Variable length

The key identifier is coded as follows.

b8	b7	b6	b5	b4	b3	b2	b1
V	RFU				Key Number		

V bit is used to denote if the corresponding key is valid or not. (0: invalid; 1: valid). The key number (by which the key is referred to in various security related operations) will be unique for all keys. There can be only up to 31 keys

in EF2. No two keys will have the same key number even if they are used for two different purposes.

The key type field provides the operations for which the key can be used. The value is coded as follows.

b8	b7	b6	b5	B4	b3	b2	b1
CC	RFU	Enc	RFU	RFU	KD	Int Auth	Ext Auth

If the CC bit is set to 1, the key can be used for computation and verification of the cryptographic checksum.

If the Enc bit is set to 1, the key can be used for symmetric encryption and decryption.

If the KD bit is set to 1, the key will be known as a master key and can only be used for derivation. A master key is not intended for direct use. In the case of a master key record, all other bits and information refer to the derived key and not for the master key.

If the Int Auth bit is set to 1, the key can be used for the internal authentication.

If the Ext Auth bit is set to 1, the key can be used for the external authentication.

If the key is to be used for the Mutual Auth, bit b1 and b2 both should be set to 1 in the key type field.

The type specific information is of variable length and is defined as per the key type field. The value is made available for each bit set to 1 in the key type field. The values are provided in the order of the bits in the key type field. Thus if the CC bit is set to 1, the type specific information will first contain the information regarding the usage of the key for the CC. The following type specification information is used.

Operation	Information
CC	None (0 bytes)
Enc	Usage Counter (2 Bytes)
KD	None (0 bytes)
Int Auth	Usage Counter (2 Bytes)
Ext Auth	Retry Counter (4 bits) and Max Retry Count (4 bits)

The following rules shall apply for the usage counters

- Usage counters for the Enc, and Int Auth are monotonically decreasing counters.
- The counter if set to ‘FFFF’, means that the counter is not used (and therefore is not changed by the usage of the key – count treated as infinity).
- Values other than ‘FFFF’ refer to the number of times that the key can

be used by the respective operations.

- The Int Auth counter will be decremented each time INTERNAL AUTHENTICATE command is used unless it is ‘0000’ or ‘FFFF’.
- The Enc counter is decremented each time a PSO:ENCRYPT or PSO:DECRYPT, SM:ENCRYPT or SM:DECRYPT operation is performed using this key as long as it is not ‘0000’ or ‘FFFF’.
- The key can be used only if the Usage Counter after decrementing (if needed) is non-zero.
- If the counters are to be decremented, they are decremented irrespective of the outcome of the operation being successful or unsuccessful.
- The initial value of the counter is set at the time of writing the record in the EF2. The value can be changed by UPDATE RECORD command if its execution is permitted by the security conditions.

The Retry Counter and Max Retry Count are coded as per the coding given for the password repository. Bits b8:b5 provide the Retry Counter value while bits b4:b1 provide the Max Retry Count. These values are used only upon the use of the EXTERNAL AUTHENTICATE/MUTUAL AUTHENTICATE commands. If the value of the Retry Counter is 0, the relevant command results in a failure. If the Retry Counter is other than 0, it is decremented by 1 (only if the Max Retry Count is not ‘F’) upon each unsuccessful authentication. Retry counter is set to Max Retry Count after each successful authentication.

The RFU byte in the key record will always be 00.

The changes in the EF2 may or may not be permitted depending upon the security attributes for that file. However this file will be referenced for validating the keys internally by the operating system.

An invalid key will result in a failure for the all commands that may use this key. However for the security conditions, an invalid key is assumed to have been satisfied. For example, if a file read operation is subject to the condition of proving the knowledge of a key which is marked invalid, the read operation shall be permitted.

The appropriate algorithm will be chosen for the operation as given in table 8.4. If the key is not permitted to be used for the operation (as per the key-type information), SW1-SW2=‘6985’ (“Condition of use not satisfied”) will be returned.

11. SCOSTA-CL Supported Commands

A SCOSTA-CL compliant ICC has two kinds of storage – volatile and non-volatile. All instructions that update the non-volatile information will execute atomically in an SCOSTA-CL compliant ICC. The effect of all such commands will be either due to the complete execution or due to no execution. The ICC may use the subsequent power up to finish the last command. The commands supported by the SCOSTA-CL are categorized in the following categories. All of these commands are appropriately referred to the ISO/IEC

documents.

- File related commands
- Security related commands
- Other commands

11.1 File Related Commands in SCOSTA-CL

SCOSTA-CL compliant OS will support the following commands as mandatory commands.

- READ BINARY
- READ RECORDS
- WRITE BINARY
- WRITE RECORD
- UPDATE BINARY
- UPDATE RECORD
- APPEND RECORD
- ERASE BINARY
- CREATE FILE
- DELETE FILE
- SELECT FILE
- ACTIVATE FILE
- DEACTIVATE FILE
- TERMINATE DF
- TERMINATE EF
- TERMINATE CARD USAGE

11.1.1 READ BINARY command

Ref: As in ISO/IEC 7816 for INS = ‘B0’.

11.1.2 READ RECORDS command

Ref: As in ISO/IEC 7816 for INS = ‘B2’.

11.1.3 WRITE BINARY command

Ref: As in ISO/IEC 7816 for INS = ‘D0’.

11.1.4 WRITE RECORD command

Ref: As in ISO/IEC 7816 for INS = ‘D2’.

11.1.5 UPDATE BINARY command

Ref: As in ISO/IEC 7816 for INS = ‘D6’.

Also see ERASE BINARY command for some details on UPDATE BINARY.

11.1.6 UPDATE RECORD command

Ref: As in ISO/IEC 7816 for INS = ‘DC’.

11.1.7 APPEND RECORD command

Ref: As in ISO/IEC 7816 for INS = ‘E2’.

11.1.8 ERASE BINARY command

Ref: As in ISO/IEC 7816 for INS = ‘0E’.

An EF is either created with the CREATE FILE command or is available in the ICC initially. The maximum size of the EF is fixed in the beginning. The EF will have the initial state of data as ‘invalid’. Data added in the EF makes its state as ‘valid’. ERASE BINARY command is used to delete the contents previously added using WRITE BINARY command. Such a command for the EF with a record structure is not needed as the record structures themselves may have a status byte (valid/invalid) and the applications can manipulate them by UPDATE RECORD command.

Depending upon the attributes of the file, ERASE BINARY command will initialize the file data to all 1’s (for write-and files) or to all 0’s (for write-or files). For write-once files, the flags shall be maintained at the level of each data-unit and it will be possible to erase data in units of one data-unit. Upon ERASE BINARY, the flags will be set to appropriate values.

The UPDATE commands will only modify the data and disregard the flags. Thus it will be possible to UPDATE data and later WRITE into a write-once file.

For maintaining the atomicity the SCOSTA-compliant OS may assume that an application shall not issue an erasure of more than 255 bytes at a time. In case, the application issues an erasure of more than 255 bytes, the command shall still be carried out even though atomicity for the command may not be guaranteed.

11.1.9 CREATE FILE command

Ref: As in ISO/IEC 7816 for INS = ‘E0’.

The CREATE FILE command in SCOSTA-CL complaint OS will require the file control parameters as given in table 3.

Table 3: File control parameters for SCOSTA-CL files

Tag	Value	Applicable to
‘80’	Size of the file in bytes	Transparent EF

‘82’	Depending upon the length	
(L=1)	File descriptor byte-FDB	DF
(L=2)	FDB + data coding byte-DCB	Transparent EF
(L=5)	FDB + DCB + MRL (2 bytes)+ #records (1 byte)	EF of records
(L=6)	FDB + DCB + MRL (2 bytes)+ #records (2 byte)	EF of records
	Note: Tag ‘82’ with length 3 and 4 is defined in ISO/IEC 7816-4 but is not used with these values of the length in the SCOSTA-CL. Tag ‘82’ with length 1 and 2 is defined in ISO/IEC 7816-4 for any file. In case of the SCOSTA-CL, these can be used only for the DF and transparent EF respectively.	
‘83’	File identifier (Mandatory field)	Any file
‘84’	DF Name	DFs
‘88’	Short EF identifier (1 ≤ SFID ≤ 30) Note: If this tag is not present, least five bits of the File identifier (tag ‘83’) be used as short EF identifier (unless they represent 0 or ‘1F’). If this tag specifies no value (length 0), the corresponding EF will have no short EF identifier. In case, multiple files are found with same SFID, the behavior of the subsequent commands with implicit selection based on SFID is undefined. In case, it results in a conflict with the key or password repository, CREATE FILE command shall fail.	EFs
‘8A’	Life Cycle Status Integer (LCSI). Default ‘05’ when not specified.	Any file
‘8C’	Security attributes in compact form	Any file
‘AB’	Security attributes in expanded form Note: If the security attributes are not defined in expanded or in compact form, default security attributes of ‘no security’ will be used. Thus the information in a file with no security attribute can be operated upon by an application.	Any file
‘7B’	SEs as defined in ISO/IEC 7816 (section 8.3 of this document). SEs may be contained in an EF in which case tag ‘8D’ is to be used to provide the file id.	DFs
‘8D’	File id of the EF under the DF containing SE templates. The SEs can be specified using ‘7B’ tag also.	DFs

The SCOSTA-CL-compliant OS will have to support P1 and P2 bytes of the command to be all 0s. Implementation with other values of the P1 and P2 is

not mandatory in an SCOSTA-CL-compliant OS. If such cases are implemented, a default value of the FCP may be assumed.

The default value for the LCSI of a file created using SELECT FILE command (if not explicitly specified in the command) will be '05' (operational state – activated).

At most one PIN repository (internal EF with SFID of '01') can be created within a DF. Similarly, at most one key repository (internal EF with SFID of '02') can be created within a DF. If an attempt is made to create a PIN repository (or a key repository) in a DF that already contains one PIN repository (or one key repository), the command should return an error.

11.1.10 DELETE FILE command

Ref: As in ISO/IEC 7816 for even INS = 'E4'.

The DELETE FILE command can be used to delete an application/sub-application in a multi-application ICC or delete an EF within a single application. For deleting a file, the following two security conditions are applicable.

- "DELETE FILE (child)" of the parent DF of the EF/DF to be deleted., and,
- "DELETE FILE (self)" of the EF/DF to be deleted.

If the DELETE FILE command is used to delete a DF, the entire sub-tree under that DF is deleted. However, this command cannot be used to delete the MF. After successful execution of this command, the parent DF of the deleted EF/DF becomes the current DF. If as a result of the execution of the command, the current DF changes, then after the execution of the command, the current EF of the card is undefined; otherwise, it remains the same as before. If the current DF is changed, the current SE and related data are also changed.

11.1.11 SELECT FILE command

Ref: As in ISO/IEC 7816 for INS = 'A4'.

It should be possible for the SELECT FILE command to return FCP of the selected file depending upon the value in parameter P2. If Le field is empty, no value will be returned by the SELECT FILE command. The ICC must indicate the number of byte available using '61XX' status. These can be retrieved by the IFD with the help of a GET RESPONSE command. When P2 is between '0C' to '0F', no value shall be returned and indication of the number of bytes using '61XX' shall not be given.

When the current DF is changed, the current SE and related data are also changed.

11.1.12 ACTIVATE FILE command

Ref: As in ISO/IEC 7816 for INS = '44'.

Similar to the SELECT FILE command, it will be possible for the application to get the FCP of the file depending upon the value in the parameter P2. The LCSI byte in the returned FCP will represent the status of the file after the execution of the command.

11.1.13 DEACTIVATE FILE command

Ref: As in ISO/IEC 7816 for INS = '04'.

Similar to the SELECT FILE command, it will be possible for the application to get the FCP of the file depending upon the value in the parameter P2. The LCSI byte in the returned FCP will represent the status of the file after the execution of the command. In case of a DF, this command will execute successfully only if all EFs and DFs that exist immediately within the DF to be deactivated are already in the deactivated or terminated state.

11.1.14 TERMINATE DF command

Ref: As in ISO/IEC 7816 for INS = 'E6'.

Similar to the SELECT FILE command, it will be possible for the application to get the FCP of the file depending upon the value in the parameter P2. The LCSI byte in the returned FCP will represent the status of the file after the execution of the command. This command will execute successfully only if all EFs and DFs that exist immediately within the DF to be terminated are already in the terminated state.

11.1.15 TERMINATE EF command

Ref: As in ISO/IEC 7816 for INS = 'E8'.

Similar to the SELECT FILE command, it will be possible for the application to get the FCP of the file depending upon the value in the parameter P2. The LCSI byte in the returned FCP will represent the status of the file after the execution of the command.

11.1.16 TERMINATE CARD USAGE command

Ref: As in ISO/IEC 7816 for INS = 'FE'.

This command is functionally the same as terminating the MF using the TERMINATE DF command and can only be executed if all EFs and DFs that exist immediately within the MF are already in the terminated state.

11.2 Security Related Commands in SCOSTA-CL

The following commands will be supported by the SCOSTA-CL compliant operating systems.

- VERIFY
- INTERNAL AUTHENTICATE
- EXTERNAL AUTHENTICATE

- MUTUAL AUTHENTICATE
- GET CHALLENGE
- MANAGE SECURITY ENVIRONMENT
- PERFORM SECURITY OPERATION
- ENABLE VERIFICATION REQUIREMENT
- DISABLE VERIFICATION REQUIREMENT
- CHANGE REFERENCE DATA
- RESET RETRY COUNTER

11.2.1 VERIFY command

Ref: As in ISO/IEC 7816 for INS = '20'.

VERIFY command in SCOSTA-CL-compliant OS will be used to verify the password with the password stored in the ICC. The reference data number as provided in P2 should be interpreted as given in table 4.

Table 4: P2 byte for the VERIFY command

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
0	0	0	0	0	0	0	0	No information is given
0	0	0	X	X	X	X	X	PIN number in the global PIN repository.
1	0	0	X	X	X	X	X	PIN# (lower 5 bits) in the local repository
All other values								RFU

When the value of P2 is 0, the PIN number shall be taken from the AT template of the currently selected SE for which the usage qualifier indicates the user-knowledge based authentication. The error conditions shall be returned as if the reference data is not found if any of the following conditions are true.

- No CRT with AT template is found in the current SE.
- Key reference is not specified in the AT template.
- Usage qualifier does not permit the user-knowledge based authentication.
- The PIN is not found in the repository or the corresponding record is an invalid one.

If the most significant bit of the PIN number is 1, then the PIN number coded in the lower five bits shall refer to the PIN in the local repository. If the most significant bit of the PIN number is 0, then the PIN number shall refer to the PIN in the global repository.

11.2.2 INTERNAL AUTHENTICATE command

Ref: As in ISO/IEC 7816 for INS = '88'.

The description for the INTERNAL AUTHENTICATE command is also valid

for the MUTUAL and EXTERNAL AUTHENTICATE commands and is indicated explicitly.

The INTERNAL/EXTERNAL/MUTUAL AUTHENTICATE command requires an algorithm reference in P1 and a key reference in P2 as per Table 5.

Table 5: P2 byte for various AUTHENTICATE commands

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
0	0	0	0	0	0	0	0	No information is given
0	0	0	X	X	X	X	X	Key Reference in Global Key Repository
1	0	0	X	X	X	X	X	Key Reference in local Key Repository
All other values								RFU

Depending upon the values of P1 and P2 for INTERNAL/EXTERNAL/MUTUAL AUTHENTICATE commands, key reference and algorithm references may be taken from the current SE, P1 or P2 as per table 6. When the references are taken from the current SE, the AT template shall be used for which the usage qualifier permits the selected operation.

Table 6: Interpretations of P1 and P2 for various AUTH commands.

P1	P2	Value of AlgoRef	Value of KeyRef
0	0	From current SE	From current SE
0	≠ 0	From current SE	P2
≠ 0	0	P1	From current SE
≠ 0	≠ 0	P1	P2

An error (‘6A88’: REFERENCE DATA not found) shall be returned when any of the following conditions are true.

- If there is no CRT with AT template in the current SE
- The key reference is not specified in the AT template of the current SE. (When the Algorithm Reference is not specified in the AT template, it shall be taken as 0). Also see section 11.2.4 for MUTUAL AUTH specific details.
- The usage qualifier in the SE does not indicate the use for the command.
- The key is not found in the key repository
- The indicated algorithm is not supported

The value of AlgoRef as indicated in Table 6 when equals 2, the INTERNAL and EXTERNAL AUTHENTICATE commands will fail and only MUTUAL AUTHENTICATE command shall be allowed to proceed to establish session key along with the authentication.

The INTERNAL/EXTERNAL/MUTUAL AUTHENTICATE commands shall return error when any of the following conditions are true (but for the exceptions as listed below).

- If the key is found in the repository but is an invalid one. In this case the behavior for the EXTERNAL AUTH command shall be as indicated in section 11.2.3. Otherwise an error code '6984' (Reference data invalidated) shall be returned.
- The key type byte in the key repository (or for the derived key in case of master key usage) does not permit the command. An error code ('6985': condition of use not valid) shall be returned.
- If the key is found to be expired (i.e. the usage counter is 0). An error code '6985' shall be returned.
- The response is not as expected for the challenge (in case of MUTUAL/EXTERNAL AUTHENTICATE).

If the indicated key reference is provided under tag '84' (key for the derived key usage) then the derived key shall be used rather than the master key itself. In this case, the key repository must also indicate the key to be a master key by the KD bit set to 1. The key must have been derived for the appropriate use (for internal authentication in case the command is INTERNAL AUTH; for external authentication in case the command is EXTERNAL AUTH; and for both in case the command is MUTUAL AUTH) using MANAGE SECURITY ENVIRONMENT command.

For the INTERNAL AUTHENTICATE command, corresponding 16-bit usage counter shall be decremented by 1 unless it is 0 or 'FFFF'.

11.2.3 EXTERNAL AUTHENTICATE command

Ref: As in ISO/IEC 7816 for INS = '82'.

The EXTERNAL AUTHENTICATE command will be subjected to the behavior as described in section 11.2.2. In addition the following shall apply.

If the body of the command is empty, SW1-SW2='63CX' (where X is the value of further allowed retries) will be returned if the record corresponding to the referenced key is valid. In this case, no counters will be modified.

If the body of the command is empty, SW1-SW2='9000' will be returned if the record corresponding to the referenced key indicated key to be invalid. No counters shall be modified in this case.

If the reference data is invalid and the body is non-empty, '6984' (REF DATA INVALIDATED) shall be returned.

When the authentication is successful (the response is as expected for the issued challenge), retry counter shall be set to the maximum retry count, otherwise the counter shall be decremented by 1.

11.2.4 MUTUAL AUTHENTICATE command

Ref: As in ISO/IEC 7816 for INS = '82'.

The MUTUAL AUTHENTICATE command will be subjected to the behavior as described in section 11.2.2. In addition the following shall apply.

The behavior of this command shall be same as an EXTERNAL AUTH followed by an INTERNAL AUTH. The behavior shall depend upon the value of the Algorithm Reference in the key repository and these may be used for the derivation of session keys as described later in section Part II.2.1.

When the MUTUAL AUTH command is used for key derivation and authentication purposes as outlined in section Part II.2.1, the command needs to perform operations related to confidentiality as well as integrity. For this purpose, the confidentiality and integrity keys shall be taken from the CT and CCT templates. In such a case, valid values of P2 must only be 0.

When the mutual auth is used only for the authentication, the key reference for this command (as given in table 6) will be valid only if the key type indicates that the key can be used for Internal Authentication as well as for External Authentication.

When the mutual auth is used for key derivation as well, the key for confidentiality must be valid for encryption and decryption (in the CT template of the SE) and the key type must permit encryption. The key for the integrity must be valid for computation and verification of crypto-checksum (in the CCT template of the SE) and the key type must permit crypto checksum. The usage counters shall be handled as usual.

The size of DATA and RESPONSE will depend upon the algorithm reference.

11.2.5 GET CHALLENGE command

Ref: As in ISO/IEC 7816 for INS = ‘84’.

The GET CHALLENGE command will return a random number that may be derived from the session counter (but not the session counter itself). This challenge can be used implicitly only for the very next command issued by the interface device. The challenge will not be valid for the subsequent commands. The quality of the random numbers generated shall be as per prevailing international standards.

11.2.6 MANAGE SECURITY ENVIRONMENT command

Ref: As in ISO/IEC 7816 for INS = ‘22’.

The MSE command will be subjected to the following.

- MSE SET command can only change the attributes of the current SE. Only those attributes that are listed as changeable in the table given below are permitted to be modified.
- MSE RESTORE command can select a specified SE as the current SE. It shall operate on the SE stored in the file or as DOs in the DF.
- MSE STORE and MSE ERASE commands shall not be permitted.

Upon a MSE SET if the component identified by tag ‘94’ is to be changed and it has a length 0 in the command body, then the previously obtained challenge

by the GET CHALLENGE command will be used as data to derive the key. For this method to work, the MANAGE SECURITY COMMAND must be the immediately next command to execute after the GET CHALLENGE command.

The SCOSTA-CL compliant OS shall have the interpretations of the CRT as given in table 7. The same interpretations will also be applicable for the secure messaging if applicable.

Table 7: Interpretation of various CRDOs in CRTs for SCOSTA-CL

Tag	Meaning	Applicable CRTs
Non-changeable components of an SE with MSE SET command		
'80'	Algorithm Reference (One byte value) (Ref. section 8.4)	AT, CT, CCT, HT
'83'	Key Reference for direct use. Use in cryptographic authentication with AT is described under INT/EXT/MUTUAL AUTH commands. For User Authentication, this tag provides the PIN identifier.	AT, CT, CCT
'84'	Key Reference for computation of session key (tags '83' and '84' to be used in mutually exclusive manner). '84' is not applicable for user auth in AT template.	AT, CT, CCT
'95'	Usage qualifier	AT, CT, CCT
Changeable components of an SE with MSE SET command		
'85'	(Length = 0). Set IV to zero. This may happen at the time of MSE RESTORE if this tag is used in the SE. (also the default value)	CT, CCT. For AT related commands, IV is always 0.
'86'	(Length = 0). Set the IV to zero and its update at the end of the operation, to the value of the last block (cn) of the cipher text.	CT, CCT
'87'	(Length = k) where k is the block size of the cipher. Set the IV to the value provided. (Length = 0). Increment the IV by 1.	CT, CCT
'94'	Data for key derivation (valid only when '84' tag is used for key reference)	AT, CT, CCT

In an SE (or in the argument to MSE SET) multiple tags for the changeable components might be specified. In that case, they shall be processed in the sequence of the occurrence. When a key is derived using tag '94', its use shall be restricted only for the operations given as an argument to the MSE SET command provided the key type of the master key permits it. If the master key is not valid for the operation for which key is being derived, MSE SET command shall return an error.

11.2.7 PERFORM SECURITY OPERATION command

Ref: As in ISO/IEC 7816 for INS = '2A'.

All the security operations relevant for the symmetric key cryptography will be applicable in the SCOSTA-CL. Therefore, computation of the cryptographic checksum, calculation of a hash code, verification of the cryptographic checksum, encipherment, and decipherment will be required in an SCOSTA-CL compliant ICC. These commands can be performed only if the security status satisfies the security attributes needed for the operation. Table 8 explains the P1 and P2 parameters specific to the PSO command. Further in this version, command chaining is not necessarily implemented. Hence reference to command chaining in the associated documents is not relevant.

Table 8: Interpretation of P1, P2 for PSO commands.

P1	P2	Meaning	Remarks
'8E'	'80'	Compute Cryptographic Checksum	Cryptographic Checksum will be at least 4 bytes. The length of the returned value will depend upon the Le field. Also see the next row below for the description on the algorithm, IV and key references.
'00'	'A2'	Verify Cryptographic Checksum	Data field will contain plain text (tag '80') and crypto checksum (tag '8E'). The CRT, for the operation will be known from the current SE. Algorithm reference will be treated as 0 in the CCT. Key reference will be present in the SE in CCT template. The IV and IV mode update shall be as described by tags '85', '86' and '87' in the CCT template. CCT shall be subjected to the key usage DO.
'86'	'80'	Encipher	Padding indicator byte will be '01' or '02' only.
'80'	'86'	Decipher	
'90'	'80'	Hash	

The Encipher and Decipher operations may also implement tag '82' and '84' for the cryptogram. These are not mandatory in SCOSTA-CL. However if tag '82' is supported, the value of plaintext for the encipher operation shall be a DO as '80' Lc DATA Padding_if_needed. The value returned by the decipher operation shall be only the value of tag '80' in decrypted DATA.

11.2.8 ENABLE VERIFICATION REQUIREMENT command

Ref: As in ISO/IEC 7816 for INS = ‘28’.

The enable verification requirement command will update the valid bit in EF1 for the record corresponding to the specified reference data. If P2 is given as 0, then the PIN identifier shall be taken from the current SE with the AT template for which the usage qualifier indicates the USER AUTH. The value of P1 can be 00 or 01. However, if the value of P1 is 00, the verification data present in the command body will be ignored. The command shall be executed only if the PIN repository permits update operation on the basis of current security status. Otherwise an error shall be returned.

11.2.9 DISABLE VERIFICATION REQUIREMENT command

Ref: As in ISO/IEC 7816 for INS = ‘26’.

This command will update the valid bit in EF1 for the record corresponding to the specified reference data. The value of the P2 shall be interpreted as given in section 0. The command shall be executed only if the PIN repository permits update operation on the basis of current security status. Otherwise an error shall be returned.

11.2.10 CHANGE REFERENCE DATA command

Ref: As in ISO/IEC 7816 for INS = ‘24’.

In order to change reference data, the command may also choose an option of providing old reference data. The reference data will be updated only if one of the following conditions is satisfied (unless the reference data is invalid).

- PIN repository permits update operation based on the current security status.
- If the old reference data is specified in the body and its verification with the stored reference data is satisfied. The security status is also updated accordingly.
- If the reference data is not specified in the body and the security status indicates an earlier satisfactory VERIFY command with reference to the same PIN.

The value of the P2 shall be interpreted as given in section 0.

11.2.11 RESET RETRY COUNTER command

Ref: As in ISO/IEC 7816 for INS = ‘2C’.

The command will modify a record in the PIN repository subject to the IFD proving the knowledge of the reference data through a VERIFY command successfully completed earlier or if the PIN repository permits update operation on the basis of the current security status. If the resetting code is provided, the retry counter is set to minimum (resetting code, Max retry count). If the resetting code in not provided, the retry counter is set to the Max retry count. The resetting code will be one byte.

The value of the P2 shall be interpreted as given in section 0.

11.3 Other Commands in SCOSTA-CL

The following commands will also be supported in the SCOSTA-CL.

- GET DATA command
- PUT DATA command
- GET RESPONSE command

11.3.1 GET DATA command

Ref: As in ISO/IEC 7816 for INS = ‘CA’.

11.3.2 PUT DATA command

Ref: As in ISO/IEC 7816 for INS = ‘DA’.

11.3.3 GET RESPONSE command

Ref: As in ISO/IEC 7816 for INS = ‘C0’.

12. Data Objects

In SCOSTA-CL compliant OS, certain data objects may be made available. These will be in compliance to the ISO/IEC 7816-6. For example, some of the relevant DOs may be used to represent the Card’s Sequence Number, Primary Account Number, and Cardholder’s Name etc. The data objects shall be accessed using GET DATA and PUT DATA commands. DOs will be attached to a DF. GET DATA/PUT DATA commands will operate on the DOs attached to the selected DF.

A SCOSTA-CL compliant ICC will have a DO with tag ‘46’ for pre-issuing data. This DO will be available in the life time of the ICC and will be used to return the chip serial number of the chip embedded in the ICC and as provided by the chip manufacturer. The embedded chips for the SCOSTA-CL compliant OS must support the way of providing a unique chip serial number which will be returned by GET DATA command. PUT DATA command for DO ‘46’ will not modify the value. The DO ‘46’ will be available at least in the MF and will have variable length. The applications can use the GET DATA command with Le=0 to find the length of the chip serial number. In response to GET DATA command with Le=0 the exact length will be returned to the application in the status bytes (SW1=‘6C’, SW2=exact length).

Part II. Cryptographic Infrastructure in SCOSTA-CL

1. Secure Messaging

The SM for the SCOSTA-CL shall be indicated by tCLA byte being ‘08’ or ‘0C’ in the tAPDU. The SM may incorporate confidentiality, integrity, both or none.

When an SM does not incorporate the integrity, the only possible value for the tCLA shall be ‘08’ and the command header will not be authenticated.

The following fields of the APDU shall be subjected to the secure messaging while computing tAPDU.

Command Header (including CLA, INS, P1 and P2).

Command Data Field (DATA)

Le field

Response Data Field (RESPONSE)

Response Status Bytes (SW1, SW2)

Under the SM, the command data field in the tAPDU (tDATA) shall be consisting of collection of the SM related DOs which can include the fields given in table 9.

Table 9: SM fields for SCOSTA-CL secure messaging.

SM Tags	*Meaning of value*	*Explanatory notes*
‘80’, ‘81’	DATA field of APDU in case of command and RESPONSE field of APDU in case of response.	
‘82’, ‘83’	Data fields provide a cryptogram. After decryption of the cryptogram, the plaintext provides the values as listed in this table (and will not include tags ‘82’, ‘83’, ‘B0’ and ‘B1’).	There shall be only a maximum of one field each of the DATA, Le, CH in the command APDU and only a maximum of one field each of the RESPONSE and SW1-SW2 in the response APDU. Thus a tag ‘80’ or ‘81’ may be found in the plaintext corresponding to tag ‘82’ or as a DO in the tDATA field. The value field of this DO is a context by itself and may include crypto-checksum (tag ‘8E’). The key and algorithm shall be from the CT template in the SM field or from the current SE (if not specified in the SM field).
‘86’, ‘87’	Padded if necessary and encrypted DATA field of the command APDU and RESPONSE field of the response APDU.	The Lc field and Le field shall be known after decryption and removal of padding as length of the data. The key and algorithms shall be taken as described for ‘82’/’83’.
‘89’	Command Header of the APDU (CLA-INS-P1-P2)	Command header may be given in the tAPDU data field. In that case, the Command Header of the tAPDU (tCLA-tINS-tP1-tP2) shall be ignored for the command execution. tCLA shall be used to formulate the

		response APDU only. If this tag is used, the integrity use is mandatory.
'8E'	Crypto-Checksum	When used, this shall be the last field in the context. If used in under the plaintext corresponding to '82' and '83' or under 'B0' and 'B1', '8E' shall be the last DO. When used in the context of the tDATA or tRESPONSE, it shall be the last field of the tDATA or tRESPONSE. The crypto-checksum shall be computed/verified using the CCT template ('B4'/'B5') provided in the SM field in the same context, or the CCT template of the current SE.
'96', '97'	Provides the value of the Le.	This SM field is valid only in the command tAPDU.
'99'	Provides the SW1 and SW2	Use of this tag requires the use of Integrity in the tAPDU. This SM field is valid only in the response tAPDU.
'B0', 'B1'	Plaintext DOs whose values include the SM fields as described in this table and shall not include '82', '83', 'B0' and 'B1' tags.	The value field is a context by itself and may include crypto-checksum (tag '8E') of relevant DOs in the value.
'B4', 'B5'	CCT template	In the context where this DO is found, the crypto-checksum is computed using the values of this DO.
'B8', 'B9'	CT template	In the context where this DO is found, the encryption/decryption is carried out using the values of this DO.

The following shall be applicable for the secure messaging.

- If the CCT and/or CT templates are not provided in the SM data fields, then the CCT and/or CT templates of the current SE shall be used.
- The DATA and RESPONSE fields may be found in one of the following places.
    - As a plaintext value with tags '80'/'81'.
    - As the decrypted and de-padded cipher-text from tags '86'/'87'.
    - While processing the DOs for the value fields of plaintext DOs 'B0'/'B1'.
    - While processing the DOs for the values fields of the decrypted

and de-padded cipher-text from tags ‘82’/‘83’.

- The Le and SW1/SW2 fields may be found in one of the following places.
    - As a plaintext value with tags ‘96’/‘97’ for Le and ‘99’ for SW1-SW2.
    - Under tags ‘82’/‘83’ whose value field when decrypted and de-padded provides collection of DOs which may include DOs with tags ‘96’/‘97’ for the Le and ‘99’ for SW1-SW2.
- The tDATA field and tRESPONSE fields of the tAPDU may include several contexts. Each context provides a collection of SM DOs. A new context may start because of tags ‘82’/‘83’/‘B0’/‘B1’. Each context may include zero or one CCT template, zero or one CT template and zero or one crypto-checksum. The crypto-checksum if present shall be the last DO in its context.
- When the CCT and CT templates are used in the SM fields, these may refer to the derived keys.
- In a context, the crypto-checksum shall be computed for all DOs which have an odd tag. The computation of the checksum shall be subjected to the padding as defined in ISO/IEC 7816-4 and to the crypto-checksum algorithms as described in this document.
- Each time an encryption/decryption is performed, the key usage counters shall be updated as defined for the PSO command.
- Keys and Algorithms used for formulating the tRESPONSE will be same as those used for corresponding SM fields in tDATA. In case, tDATA did not carry any encryption but tRESPONSE need to carry data, the key and algorithms shall be taken from the current SE. If there is no CT template in the current SE, the response shall be carried in plaintext using tags ‘80’ or ‘81’ in such cases.

2. Session Key Derivation

SCOSTA-CL will maintain at least two session keys, one for the confidentiality and another for the integrity. In case the derived keys are to be used for the authentication purposes as well, these may share space with the session keys for the confidentiality (i.e. there may be only one derived key for authentication and confidentiality). When the keys are derived, the OS shall have to maintain the usage for which the key is derived and permit only that use.

It will be possible for a SCOSTA-CL compliant operating system to be able to derive the session keys using one of the following methods.

2.1 Authentication along with Session Key Derivation

Session key can be derived along with the authentication using GET CHALLENGE and MUTUAL AUTHENTICATE command. This has been

described in section 4.3 in more details when the Algorithm Reference is 0 or 1. Two keys shall be derived in this manner, one for the confidentiality and another for the integrity. The confidentiality key shall be usable for encryption and decryption (this can be restricted further by the CRT usage tag '95' in the CT template in the current SE). The integrity key shall be usable for the computation and verification of cryptographic checksum (this can be restricted further by the CRT usage tag '95' in the CCT template).

2.2 Session Key Derivation using MANAGE SECURITY ENVIRONMENT Command.

The MANAGE SECURITY ENVIRONMENT can be used for the key derivation of the session keys. For this tag '94' in the appropriate CRT shall be used. For example, if the CT template of the current SE is selected, the key shall be derived for the confidentiality as per the key usage qualifier in that SE. The key derivation method shall use the tag '84' (in the appropriate CRT) to find the key reference of the master key. It shall also check if the key is permitted as a master key in the key repository. The derived key shall be taken as a 3DES encryption of 16-byte data (tag '94') using a 16-byte master key. For this encryption, the IV shall be taken as 0 (and it shall not modify the IV for confidentiality/integrity maintained separately by the OS). The encryption mode will be CBC as shown in figure 2. The resulting 16-byte number shall be treated as a 3DES key. The key shall be subjected to the key usage based on the key type parameter in the key repository. Thus a key derived using the CT template for example, can be made to work only for encryption or only for decryption or for both.

2.3 Key Derivation for Authentication

Keys may be derived for the authentication purposes if the corresponding master key reference is provided in the AT template in the current SE. The algorithm for the key derivation is the same as that for the session key derivation using MANAGE SECURITY ENVIRONMENT command as described in section 2.2.

3. Setting Initial Value (IV) for Cryptographic Methods

Many cryptographic algorithms require an initial value (IV) size of which usually depends upon the algorithm.

Each cryptographic operation may in turn update the IV at the end. There are the following possibilities for the IV value and IV update.

The value of the IV is set as per the details given in section Part I.11.2.6. IV is also updated at the end of the cryptographic operation as given in section Part I.11.2.6.

Initial value of the IV may also be determined the way the keys are derived as defined in section 4.3.

SCOSTA-CL compliant OS shall maintain two different IVs and IV update modes, one each for the confidentiality and integrity. IV for integrity shall be

updated at the end of computation/verification of cryptographic checksum and IV for confidentiality shall be updated at the end of encryption/decryption by means of a PSO command or by means of a secure messaging.

4. Cryptographic Algorithms in SCOSTA-CL

A SCOSTA-CL compliant OS shall support all cryptographic algorithms as described in Part I.8.4. In this section, the algorithms are described in details.

4.1 Algorithms for Confidentiality

Algorithms for the confidentiality include algorithms for Encryption and Decryption. The SCOSTA-CL compliant OS shall support at least the 3DES algorithm. This algorithm shall be subjected to padding (as defined in ISO9797-1 padding method 3) as well as CBC mode of encryption/decryption.

The algorithm reference of 0 and 1 in the CT template in a CRT shall mean the 3DES algorithm as described here.

The 3DES basic block encryption shall use a 16-byte key data structure as per ISO 11568-2 standard. This shall be considered as two 8-byte wide keys for DES algorithms. The IV for the algorithm and the method to update the IV at the end of encryption/decryption can be set by MANAGE SECURITY ENVIRONMENT command.

Only one IV need be maintained by the OS for the confidentiality which shall be different from the IV maintained by the OS for the integrity.

The 3DES algorithm is explained pictorially in figures 1 and 2.

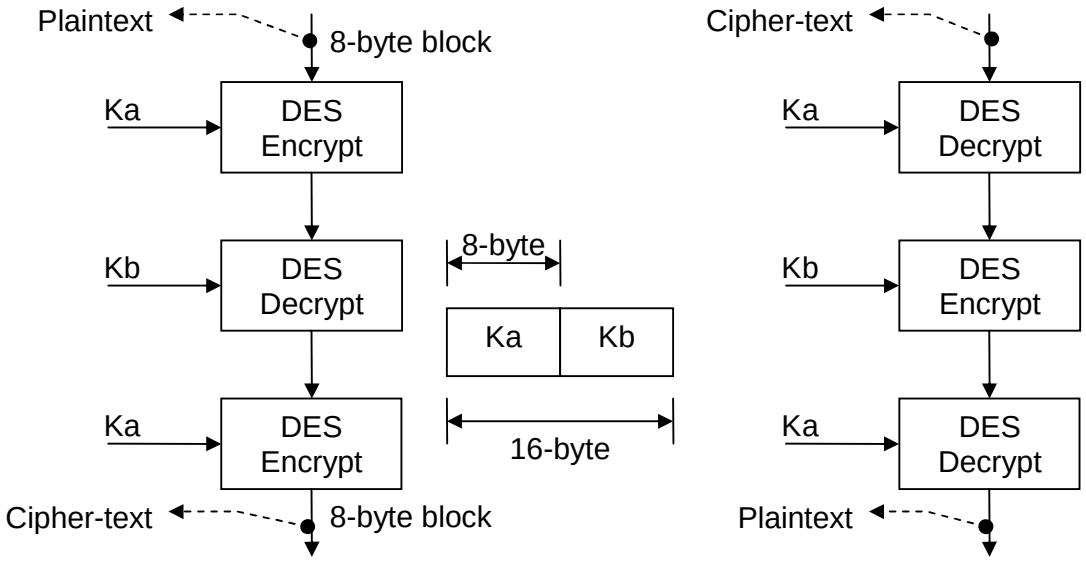

Figure 1: Basic encryption and decryption mechanisms in 3DES algorithm

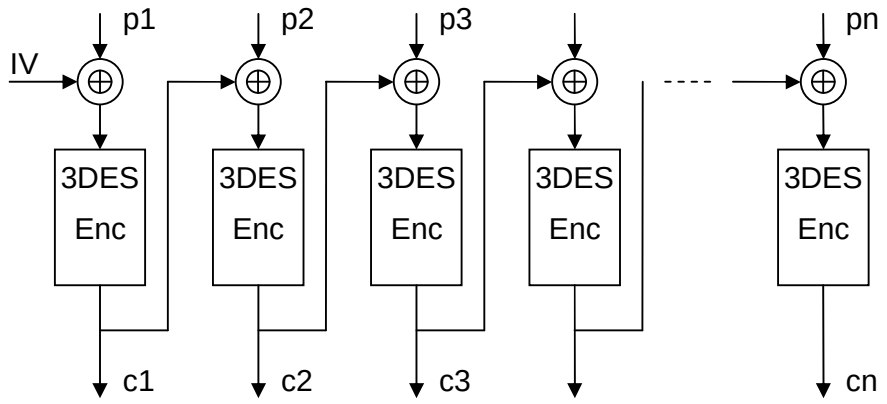

Figure 2: 3DES Encryption of large messages using CBC mode.

Depending upon the value of the IV related tags, the new value of the IV may be set automatically to cn (the last cipher-text block) when the chaining mode is used for the IV.

The IV must be initialized whenever the current SE is changed (for example, because of SELECT FILE command or by MANAGE SECURITY ENVIRONMENT command).

4.2 Algorithms for Integrity

The SCOSTA-CL compliant OS shall support two algorithms for the integrity. The first algorithm (known as 3DES based CBC residue) shall be indicated by algorithm reference of 0 in the CCT template of the CRT. ISO/IEC 9797-1 algorithm 2 for MAC shall be indicated by algorithm reference of 0 or 2.

The algorithms may need an IV which shall be taken as IV for the integrity.

4.2.1 CBC Residue based integrity

In this mode, the 3DES algorithm shall be used for large messages. The integrity code (or cryptographic checksum) shall be derived from the cn (the last block of the encryption as shown in Figure 2).

It is recommended that an application should not use identical keys for confidentiality and integrity.

The cryptographic checksum (or the integrity code) shall be at least 4 bytes long and at most 8 bytes long (the size of the cn in 3DES). In case, smaller than 8 bytes are used for the integrity code, those many bytes from the least significant bytes onward shall be returned out of the CBC residue. The IV and the method to update IV shall be governed by the appropriate tags in the CCT template of the CRT. The operations will be similar to the ones described earlier.

4.2.2 ISO/IEC 9797-1

This algorithm shall be indicated by the algorithm reference of 02 in the CCT

template of the CRT to a SCOSTA-CL compliant OS. In this mode, IV shall be first incremented by 1 and shall not be updated at the end of the computations.

This algorithm is represented in figure 3.

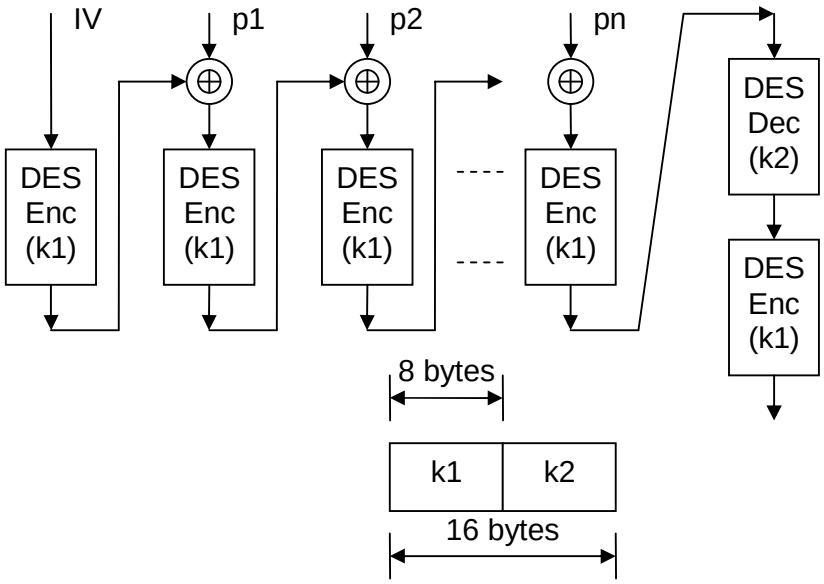

Figure 3: ISO/IEC 9797-1 based Integrity Algorithm.

4.3 Algorithms for Authentication

There shall be at least two algorithms for cryptographic authentication (INTERNAL, EXTERNAL and MUTUAL AUTH).

The algorithm reference of 0 or 1 shall refer to the 3DES based challenge response method. In this method, a challenge of 8 bytes shall be given by the challenger (IFD or ICC) to the subject (ICC or IFD). The subject shall encrypt the challenge using the indicated key and provide 8-byte response obtained as cipher-text from the 3DES encryption as described in figure 1.

The algorithm reference of 2 shall refer to the 3DES based authentication as well as session key establishment of ISO/IEC 11770-2 key establishment mechanism and as described in figure 4.

4.4 Algorithm for Hash

The SCOSTA-CL compliant OS shall use the SHA-1 algorithm as described in FIPS-140 for message digest operations when an algorithm reference of 00 and 01 are used in the HT template. The maximum size of the data to be hashed shall not exceed 255 bytes.

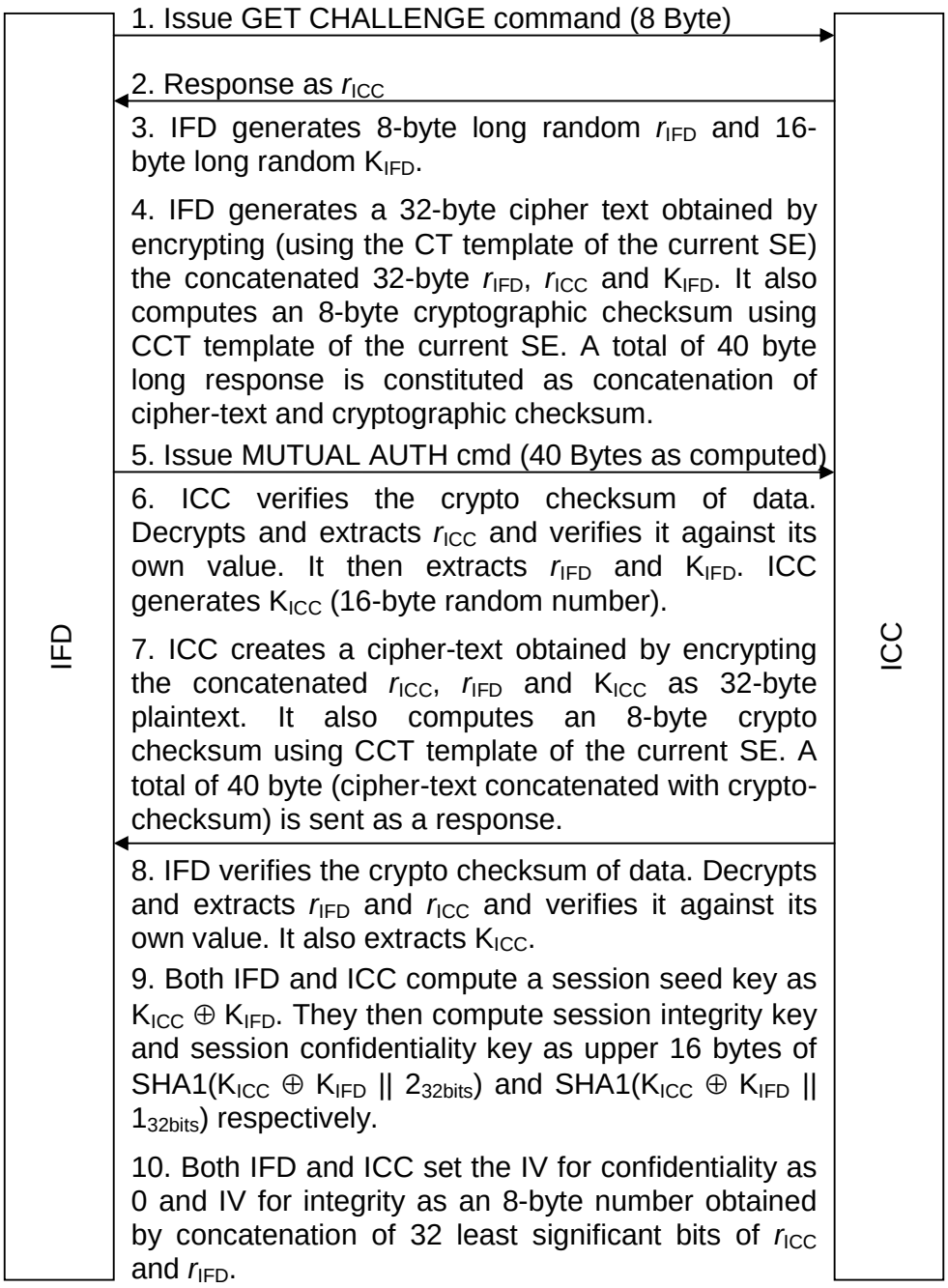

Figure 4: Authentication and key establishment protocol.