TS 101 206-7 V1.2.1 (1998-01)

Technical Specification

Identification card systems; Telecommunications IC cards and terminals; Part 7: Security module

European Telecommunications Standards Institute

Reference
RTS/PTS-00012 (b61r0ior.PDF)

Keywords
Card, security

ETSI Secretariat

Postal address
F-06921 Sophia Antipolis Cedex - FRANCE

Office address
650 Route des Lucioles - Sophia Antipolis
Valbonne - FRANCE
Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16
Siret N° 348 623 562 00017 - NAF 742 C
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° 7803/88

X.400
c= fr; a=atlas; p=etsi; s=secretariat

Internet
secretariat@etsi.fr
http://www.etsi.fr

Contents

Intellectual Property Rights

IPRs essential or potentially essential to the present document may have been declared to ETSI. The information pertaining to these essential IPRs, if any, is publicly available for **ETSI members and non-members**, and can be found in ETR 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available **free of charge** from the ETSI Secretariat. Latest updates are available on the ETSI Web server (http://www.etsi.fr/ipr).

Pursuant to the ETSI Interim IPR Policy, no investigation, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETR 314 (or the updates on http://www.etsi.fr/ipr) which are, or may be, or may become, essential to the present document.

Foreword

This Technical Specification (TS) has been produced by ETSI Project Pay Terminals and Systems (PTS).

TS 101 206-7 version 1.1.1 is the handover version to Comité Européen de Normalisation (CEN) for becoming EN 726-7.

TS 101 206-3, 4 and 7 version 1.2.1 represent an update of the documents handed over to CEN for becoming EN 726-3, 4, and 7. PTS has used version 1.2.1 rather than the handover version to CEN (version 1.1.1) for producing the conformance testing specification for EN 726-3, 4 and 7.

ETSI has produced a set of TSs which are not a copy of any CEN published EN. The TSs are complete and consistent documents with references among themselves. It has been made clear in these TSs that they are contributions to the CEN work for publication as EN (after re-editing the references). Once published by CEN as EN, ETSI will withdraw its TSs.

History of EN 726

EN 726 was prepared by ETSI Technical Sub-Committee (STC) Terminal Equipment 9 (TE9), adopted by CEN/TC 224 and submitted to the CEN formal vote.

EN 726 consists of seven parts covering Identification card systems; Telecommunications Integrated Circuit (IC) cards and terminals; as identified below:

Part 1: "System overview";

Part 2: "Security framework";

Part 3: "Application independent card requirements";

Part 4 "Application independent card related terminal requirements";

Part 5: "Payment methods";

Part 6: "Telecommunication features";

Part 7: "Security module".

Overview of ETSI deliverables on EN 726 family

TS 101 206-1	"Identification card systems; Telecommunications IC cards and terminals; Part 1: System overview".
TS 101 206-3	"Identification card systems; Telecommunications IC cards and terminals; Part 3: Application independent card requirements".
TS 101 206-4	"Identification card systems; Telecommunications IC cards and terminals; Part 4: Application independent card related terminal requirements".
TS 101 206-7	**"Identification card systems; Telecommunications IC cards and terminals; Part 7: Security module".**

Overview of ETSI deliverables on EN 726 conformance testing family

TS 101 203-1/2/3	"Identification card systems; Telecommunications IC cards and terminals; Test methods and conformance testing for EN 726-3".

TS 101 204-1/2/3	"Identification card systems; Telecommunications IC cards and terminals; Test methods and conformance testing for EN 726-4".
TS 101 207-1/2/3	"Identification card systems; Telecommunications IC cards and terminals; Test methods and conformance testing for EN 726-7".

1 Scope

The present document specifies:

a) the minimum security requirements for a Security Module (SM);

b) the general card related functions embedded in the SM - terminal protocols including minimum data exchange. The data elements and cryptographic processing described in annex A for the case where the SM is an IC card shall be supported if the SM is not an IC card or the configuration of the system, e.g. where the SM handles more than one terminal/user card. Configurations where the SM handles more than one terminal/user card are not fully described in the present document. Further mechanisms may be required to enable these configurations;

c) the necessary security services and mechanisms to provide application and cryptographic security information for the processing of telecommunication transactions;

considering several types of SM at different locations in the system and different system configurations.

The present document allows interaction between the IC card and the SM via the terminal in a way that may be functionally transparent to the terminal and possibly also the network(s). This provides the flexibility to use different techniques, commands and message formats at the terminal - SM interface.

The present document supports IC cards with payment applications following EN 726-5 [3] and all card applications following clauses 4 to 7 of the present document. The present document supports IC cards following TS 101 206-3 [2]. Other telecommunication IC cards which do not follow EN 726, e.g. simple memory cards, may also be supported by the SM described in the present document.

2 References

References may be made to:

a) specific versions of publications (identified by date of publication, edition number, version number, etc.), in which case, subsequent revisions to the referenced document do not apply; or

b) all versions up to and including the identified version (identified by "up to and including" before the version identity); or

c) all versions subsequent to and including the identified version (identified by "onwards" following the version identity); or

d) publications without mention of a specific version, in which case the latest version applies.

A non-specific reference to an ETS shall also be taken to refer to later versions published as an EN with the same number.

[1] EN 726-2: "Identification Card Systems - Telecommunications Integrated Circuit Cards and Terminals - Part 2: Security framework".

[2] TS 101 206-3: "Identification card systems; Telecommunications IC cards and terminals; Part 3: Application independent card requirements". To become EN 726-3.

[3] EN 726-5: "Identification Card Systems - Telecommunications Integrated Circuit Cards and Terminals - Part 5: Payment methods".

[4] ISO/IEC 7816: "Identification cards - Integrated circuit(s) with cards contacts -

Part 1: Physical characteristics (1987)".

Part 2: Dimensions and locations of the contacts (1988)".

Part 3: Electronic signals and transmission protocols (1990)".

Part 4: Inter industry commands for interchange (1995)".

Part 5: Numbering system and registration procedure for application identifiers".

Part 6: Interindustry data elements".

[5] ISO/IEC 10202: "Financial Transactions cards - Security architecture of financial transaction systems using integrated circuit cards

Part 4: Secure application module".

Part 5: Use of algorithms".

[6] ISO 4217: "Codes for the representation of currencies and funds".

3 Definitions and abbreviations

3.1 Definitions

For the purposes of the present document, the following definitions apply:

Access Conditions (AC): A set of security attributes associated to a file.

application: An application consists of a set of security mechanisms, files, data and protocols (excluding transmission protocols) which are located and used in the IC card and outside the IC card (external application).

Application Provider (AP): The entity which is responsible for the application after its allocation. One AP may have several applications in one card. The files allocated in the card corresponding to one application are called a card-application. There may exist several applications on a given card from the same application provider or from several application providers.

application session: A link between the card application part and the external world application part of the same application.

Application Specific Command set (ASC): To a DF can be associated, optionally, an ASC-set (an application specific command set and/or an application specific program). This means that when selecting this application, the general command set is extended or modified by this specific command set. The ASC is valid for the whole subtree of this application unless there are other ASCs defined at the lower levels of this application.

card: A multi-application card can be considered as a set of files, some of them shared by the different application providers and/or the card issuer, other files owned exclusively by the different application providers or the card issuer. Files can, e.g., be read, written or executed.

Dedicated File (DF): A file containing AC and allocatable memory. It may be the parent of elementary files and/or dedicated files.

Elementary File (EF): A file containing AC, data or a program and no other files.

EF_{DIR} is an elementary file at the MF or at DF level, which contains a list of all, or part of, available applications in the card (see also ISO 7816-5 [4]).

EF_{ID} is an elementary file at the MF level, containing the identification number of the card.

EF_{ICC} is an elementary file at the master file level, containing general information concerning the IC and IC card.

EF_{KEY_OP} is an elementary file containing operational keys.

EF_{KEY_MAN} is an elementary file containing management keys.

EF_{DIK_OP} is an elementary file containing diversified operational keys.

EF_{DIK_MAN} is an elementary file containing diversified management keys.

file IDentifier (ID): Each file (MF, DF, EF) has an identifier consisting of 2 bytes.

key qualifier: An unambiguous set of data to select a specific keyset in the SM.

Master File (MF): The mandatory unique file representing the root of the file structure and containing AC and allocatable memory. It may be the parent of elementary files and/or dedicated files.

operating system: That which is required to manage the logical resources of a system, including process scheduling and file management.

path: Concatenation of file identifiers without delimitation.

secrets: Algorithm(s), related key(s), security procedures and information.

Security Module (SM): A device containing logically and physically protected secrets - algorithm(s), related key(s), security procedures and information to protect applications in such a way that unauthorized access is not feasible. In order to achieve this the module may in addition be further physically, electrically and logically protected.

security module provider: See ISO/IEC 10202-4 [5].

sequence control: A feature assuring that operations, invoked by commands, can only be performed by the SM in allowed, predefined order(s).

3.2 Abbreviations

For the purposes of the present document, the following abbreviations apply:

AC	Access Condition
AID	Application IDentifier
AUT	Authenticated
BCD	Binary Coded Decimal
CEN	Comité Européen de Normalisation
CHV	Card Holder Verification information
DF	Dedicated File
EF	Elementary File
EN	European Norm
EW	External World
IC	Integrated Circuit
ID	Identifier of a file
IFD	InterFace Device
INT.AUTH	Internal Authentication
INV	Invalidate
LEN	Length
MAC	Message Authentication Code
MF	Master File
NEV	Never
PIN	Personal Identification Number
PRO	Protected
RAN	Random
REH	Rehabilitate
RFU	Reserved for Future Use
SM	Security Module
STC	Technical Sub-Committee of ETSI
TC	Technical Committee
TE9	Terminal Equipment 9 (Technical Sub-Committee, STC, of ETSI)
UC	User Card

4 Physical characteristics of the SM

If the SM is an IC card the physical characteristics shall be in accordance with TS 101 206-3 [2], clause 4.

For any other case the physical characteristics are out of the scope of the present document.

5 Electronic signals and transmission protocols

If the SM is an IC card the electronic signals and transmission protocol(s) shall be in accordance with TS 101 206-3 [2], clause 5.

For any other case the electronic signals and transmission protocol are out of the scope of the present document.

6 Logical model for SM

A SM following the present document contains:

a) permanent secrets (master key(s));

b) temporary secrets (diversified key(s));

c) a balance (optional);

d) an operating system to handle secure access to the secrets mentioned above. Additional options are possible as long as they do not interfere with the basic contents, but are out of the scope of the present document.

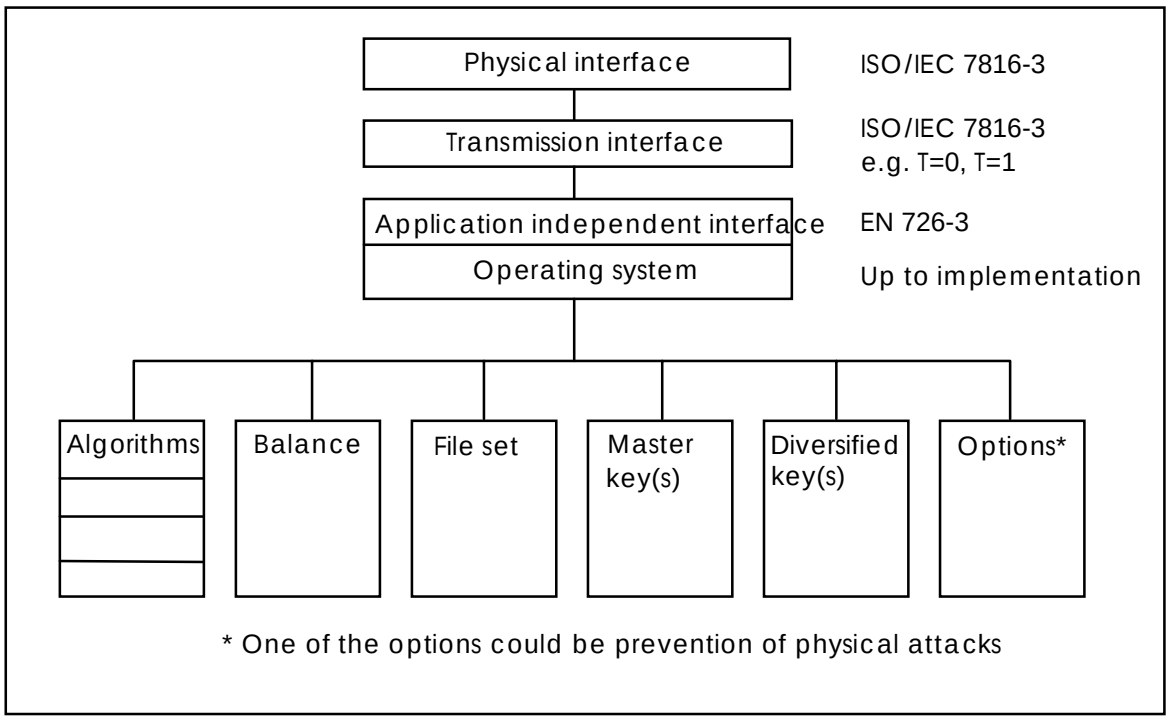

Figure 1: General structure of a SM

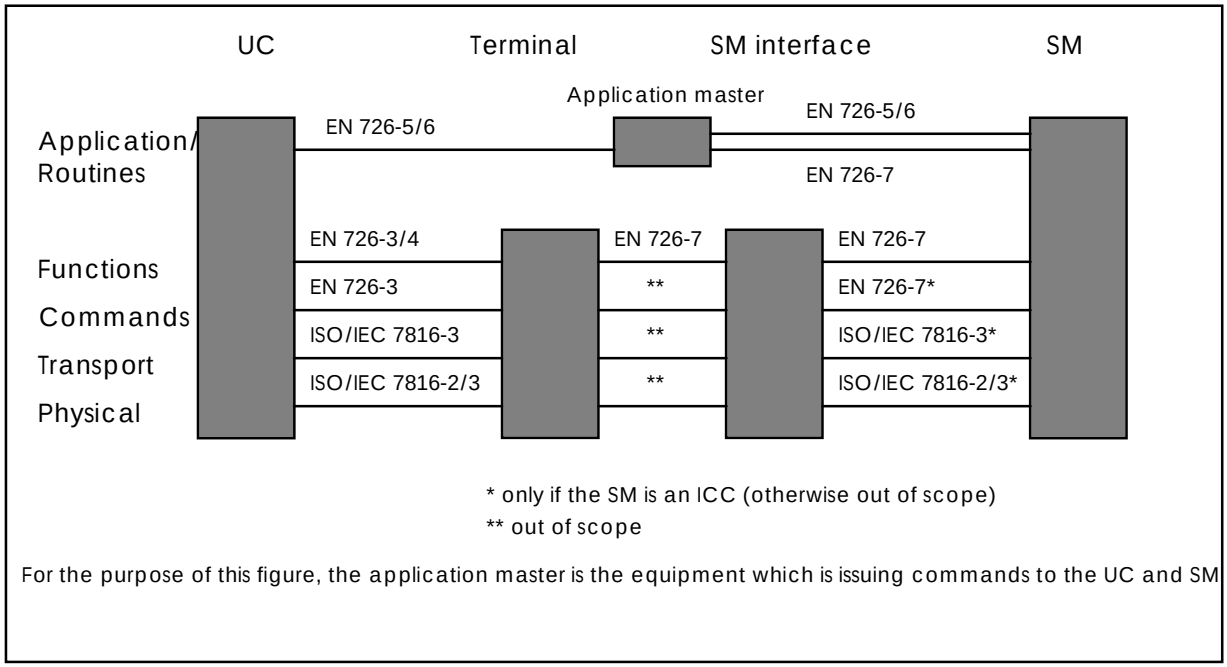

Figure 2: Standardized communication between a UC and a SM

7 General concepts

7.1 General security principles

The SM provider shall be responsible for the life cycle of the SM.

Any aspect in the operation of a SM or data obtainable from a SM shall NOT compromise the security of the system.

Cryptographic keys used in the SM and the UC shall be managed in such a way that the security of any system using SMs or UCs is not compromised.

During the application life cycle the need to change keys in the UC may appear. To support this the SM may contain several keyset versions.

Interoperability of telecommunication SMs in Europe is based on the exchange of secret keys. The mechanisms for downloading keys into the SM are not describe in the present document. The isolation of the key management systems between different operators requires the use of a trusted party.

7.1.1 Access Conditions (AC)

For the functions described in this part of the standard no ACs have to be fulfilled in the SM, but the usage of the keys may be restricted. The SM uses diversified keys for functions in the UC where the ACs require a key operation. For the management of the SM by the external world, the ACs defined in TS 101 206-3 [2] shall apply only if the SM is an IC card.

The SM may also perform a key diversification for application specific functions in the UC which are not defined in TS 101 206-3 [2]. In this case, the SM shall restrict the usage of the key(s) to these specific function(s).

7.1.2 Sequence control

A sequence control can be achieved by using ACs of the UC on the SM. Sequence control functions where the SM checks if it is allowed to perform commands and status responses can optionally be added if needed in the application. The implementation is out of the scope of the present document. Nevertheless the following applies, if the command is allowed the operation will be performed and the results will be the outcome of the command. If the command is not allowed the operation will not be performed and the status conditions will contain "Command out of sequence". The internal reaction of the SM is application dependent.

7.2 SM Life cycle

See ISO/IEC 10202-4 [5].

7.3 Configuration of the system

The general card related functions at the SM - terminal interface including minimum data exchange.
The user card - terminal interface shall not be affected by type of implementation of the SM.

Four possible configurations of the SM are shown in figures 3 to 6.

In the interests of completeness all four locations are shown. However, it is probable that for the more distant locations a higher level transport protocol may be appropriate. Such high level transport protocols are outside the scope of the present document.

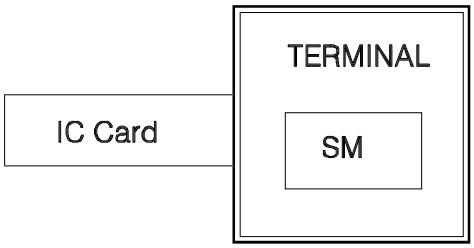

Figure 3: SM included in the TERMINAL

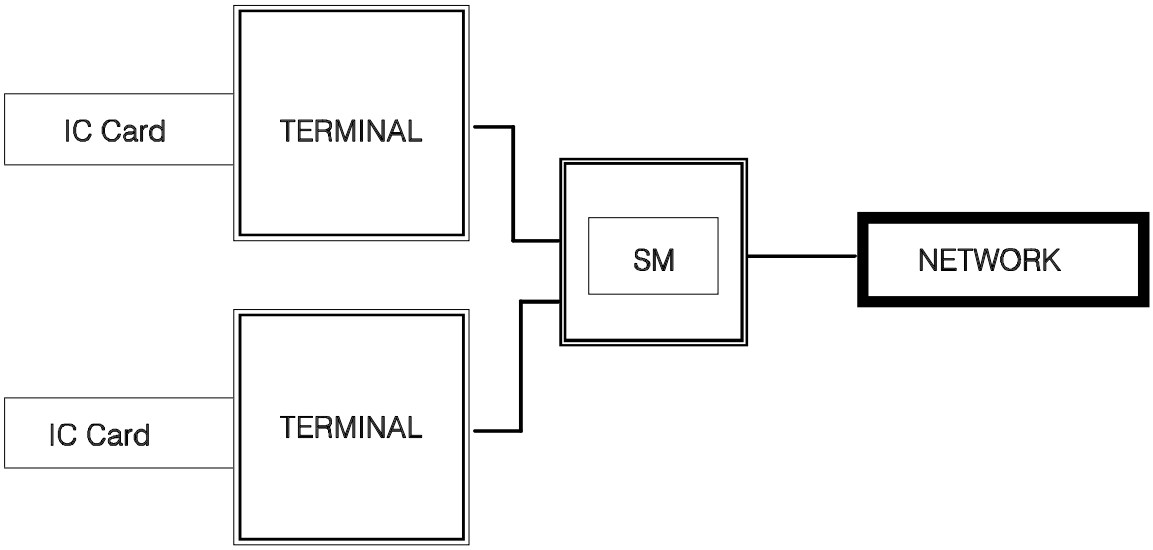

Figure 4: SM included in a local concentrator

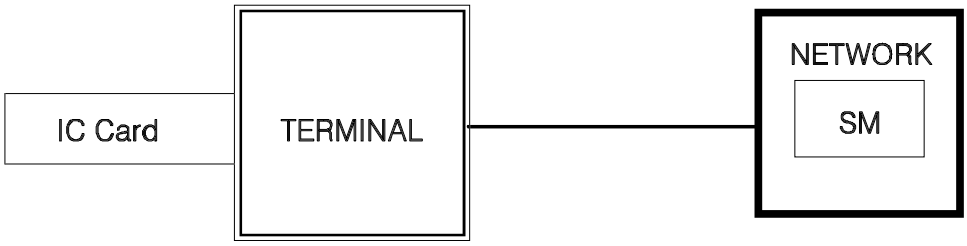

Figure 5: SM in the network

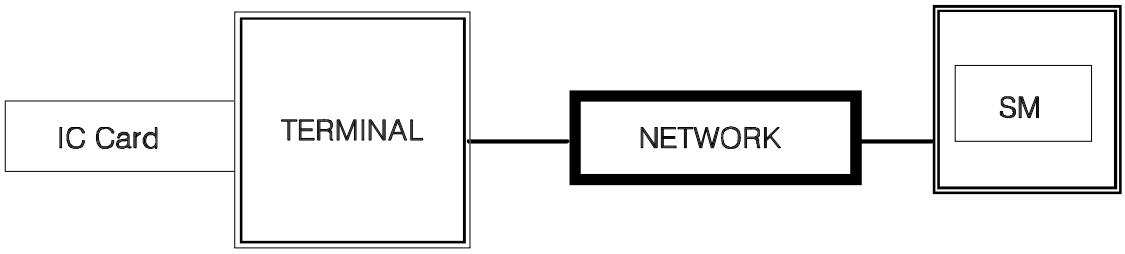

Figure 6: Remote SM

A distinction between normal operations (transfer of sensitive data such as payment transactions) and more sensitive operations (such as creation of sensitive data) may be necessary. In this case, two kinds of SM may exist. SM1 situated in an open environment only allowing normal operations and SM2 (and terminal) located in a secure environment and allowing in addition e.g. personalization.

For SMs covering more sensitive operations, specific security tools are necessary as physical protection.
These requirements are out of the scope of the present document.

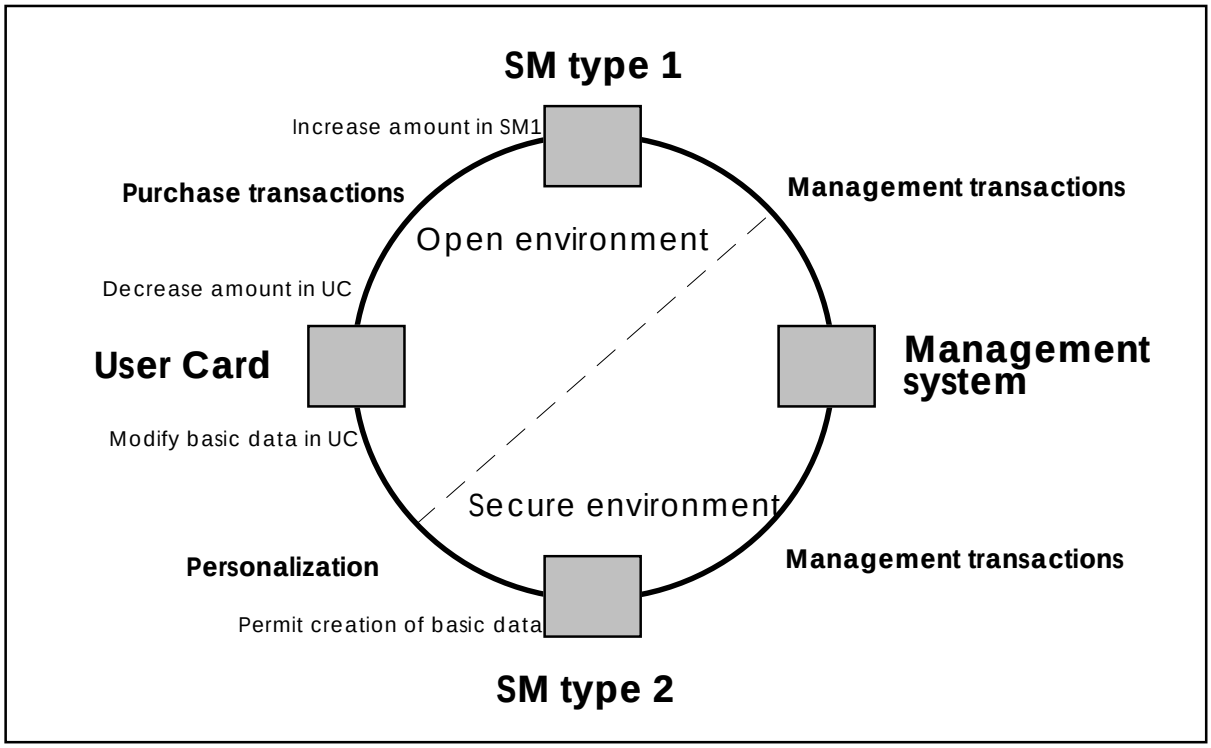

Figure 7: An example of a system configuration

7.4 SM security functions

A SM contains:

a) secrets which allow the terminal to access protected parts of the UC;

b) secrets which allow the authenticity of the UC to be checked;

c) optionally, a balance that can be increased and decreased in a secure manner in the SM, coupled with a corresponding reverse operation in the UC.

A SM shall be logically and physically protected against attacks aimed at exposing or abusing secrets during the whole life cycle of the SM.

All functions shall be independent of transmission protocols.
These may be different at the terminal - UC and at the terminal - SM interface.

In order to fulfil the AC = Authenticated (AUT) or Protected (PRO) of files in the User Card (UC), any related functions in the UC and the SM shall use the same keys and algorithms.

For the purpose of the present document the functionality is based on:

a) symmetric algorithms;

b) diversified keys.

However, the use of asymmetric or hybrid crypto systems may be defined later.

8 Description of the functions of a SM

This clause only describes the additional specific functions necessary for a SM. If the SM is an IC card the remaining part of this clause and annex A apply. Additional functions to support other applications outside the scope of EN 726 are not precluded. Any such function shall not conflict with the content of EN 726. The command header and trailer referred to in this clause are data elements used to control the processing of data by the SM in accordance with the function specified.

8.1 Functions without MAC

8.1.1 SELECT KEYSET

The purpose of this function is to unambiguously address a specific keyset in the SM, that shall be used at least for the next cryptographic computation, corresponding to the specific command, file, AC and application of the UC.
In case a UC following the present document is used, the SM shall select a keyset corresponding with the key file version indicated by the UC.

Input: a/ command header;

b/ key qualifier.

Table 1: Key qualifier (in case of a MF of the UC)

Bytes	Description	Length
1 - 4	IC card manufacturing references (coded according to TS 101 206-3 [2])	4 bytes
5	Card personalizer ID (coded according to TS 101 206-3 [2])	1 byte
6 - 7	File ID of the EF_{KEY}	2 bytes
8	Keyfile version (coded according to TS 101 206-3 [2])	1 byte

Table 2: Key qualifier (in case of a DF in the UC)

Bytes	Description	Length
1 - X	Application IDentifier (AID) (coded according to TS 101 206-3 [2])	1 - 16 bytes
(X+1) - (X+2)	File ID of the EF_{KEY}	2 bytes
X+3	Keyfile version (coded according to TS 101 206-3 [2])	1 byte

Output: a/ data: none;

b/ status.

8.1.2 DIVERSIFY KEYSET

The purpose of this function is to derive temporary keys from all the master keys contained in the previously selected keyset using application specific diversification data. The temporary keys are written in a specific temporary file. After key diversification, this temporary file becomes relevant for the DF in the UC.

This diversified keyset is kept in the SM and is valid until the next diversification using the same master keyset takes place.

NOTE: To increase the prevention of misuse of diversified keys sequence control can be used.

All secret keys, diversified as well as master keys, shall never (NEV) be allowed to be read out from the SM.

Pre-conditions: before the DIVERSIFY KEYSET function is performed, the relevant keyset shall be selected by use of the SELECT KEYSET function.

Input: a/ command header;

b/ diversification data;

c/ algorithm ID.

Table 3: Diversification data

Bytes	Description	Length
1 - X	Diversification data (application dependent)	1 - 16 bytes

Output: a/ data: none;

b/ status.

8.1.3 ASK PARAMETER

The purpose of this function is to transfer a parameter from the SM to the terminal.

This parameter is given to the terminal and used inside the SM to compute the next incoming cryptogram.

This parameter shall be valid until the next function requiring a challenge is performed in the SM except in the case of VERIFY MAC. In the case of VERIFY MAC only, the parameter shall remain valid for a subsequent function.

Types of parameters:

a) Challenge (random);

b) Challenge (counter).

Several counters for providing challenges, related in different keysets, may exist. In case of multiple counters the counter related to the previously selected keyset shall be used to provide the challenge, else a common SM-counter shall be used.

NOTE 1: It is proposed not to specify the counter structure for the case the SM is an IC card in order to allow flexibility for the way the works.

NOTE 2: For some applications a counter, e.g. a transactions number, may be used instead of a random number. In case the SM not being an IC card functionality shall be equivalent to what is described in subclause A.5.3.1.

Input: a/ command header;

b/ type of parameter

c/ length.

Output: a/ parameter;

Table 4: Parameter

Bytes	Description	Length
1 - X	Challenge	X bytes

b/ status.

8.2 Functions used to compute a cryptogram or MAC

If TESA 7 is used, a cryptogram is computed according to TESA 7 mode "authentication" (see EN 726-2 [1], subclauses A.3.1 and A.3.2) and MAC is computed according to TESA 7 mode Message Authentication Code ("MAC") (see EN 726-2 [1], subclauses A.2.3 and A.2.4).

NOTE: In other parts of the present document the result of the MAC computation may be also referred as "cryptogramm".

8.2.1 COMPUTE LOAD KEY

The purpose of this function is to provide an enciphered key and the related cryptogram to the UC. The SM shall perform an encipherment of the key (indicated in the input parameters) of the previously selected keyset. The key used to perform the encipherment shall be indicated in the command header of COMPUTE LOAD KEY. The enciphered key, the contents of the data field and the challenge previously given by the UC shall be used to provide a cryptogram.
The input data for this function contains the necessary parameters used in the UC for verifying the cryptogram.

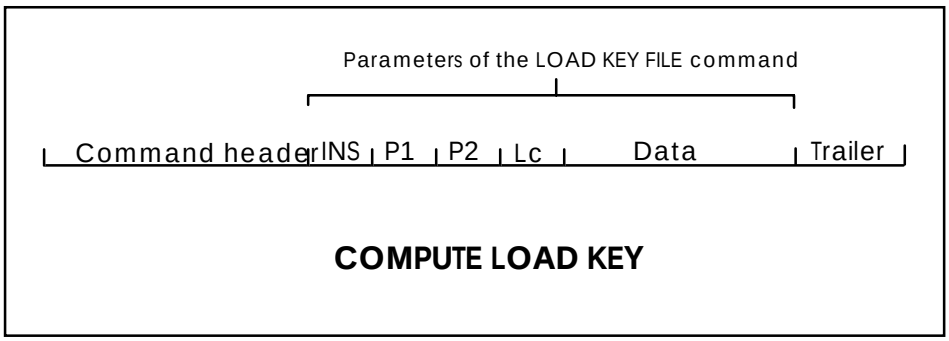

Figure 8: General structure of COMPUTE LOAD KEY

Pre-conditions: the challenge used to compute the cryptogram shall be obtained from the UC by an ASK RANDOM function and transmitted by the terminal to the SM using a GIVE RANDOM;

the keyset shall be selected using the SELECT KEYSET function. The relevant key file shall be the one containing the keys after diversification.

Input: a/ command header/trailer (COMPUTE LOAD KEY);

b/ data (parameters of the following UC command):

Table 5: Data

Bytes	Description	Length
1	INS byte of the following LOAD KEY FILE command sent to the UC (Coded according to TS 101 206-3 [2])	1 byte
2	P1 of the following LOAD KEY FILE command sent to the UC (coded according to TS 101 206-3 [2])	1 byte
3	P2 of the following LOAD KEY FILE command sent to the UC (coded according to TS 101 206-3 [2])	1 byte
4	L_c of the following LOAD KEY FILE command sent to the UC (coded according to TS 101 206-3 [2])	1 byte
5	Key file version (only present if key number 0)	1 byte
6	Key length	1 byte
7	Algorithm identifier	1 byte

c/ key number (indicated in the command header of the LOAD KEY command (P2));

d/ EF_{KEY} type (EF_{KEY_MAN} or EF_{KEY_OP}) indicated in the command header of the LOAD KEY FILE command (P1).

Output: a/ response data;

Table 6: Response data

Bytes	Description	Length
1 - 16	Enciphered key	16 bytes
17 - 24	MAC	8 bytes

b/ status.

8.2.2 COMPUTE MAC

For functions in the UC which need a MAC in order to fulfil the ACs, the general function COMPUTE MAC shall be used in the SM to calculate the MAC. The input data for this function contains the necessary parameters used in the UC for verifying the MAC (command header and data).

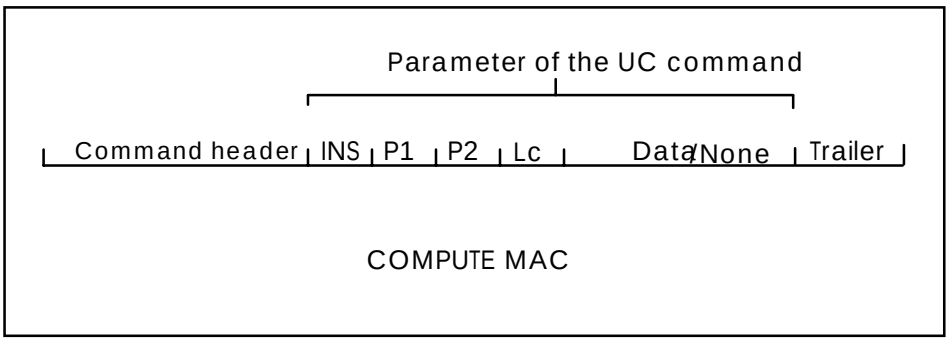

Figure 9: General structure of COMPUTE MAC

Pre-conditions: the challenge used shall be obtained from the UC by an ASK RANDOM function and transmitted by the terminal to the SM using a GIVE RANDOM;

the keyset shall be selected using the SELECT KEYSET function. The relevant key file shall be the one containing the keys after diversification.

Input: a/ command header/trailer (COMPUTE MAC);

b/ data (parameters of the following UC command):

Table 7: Data

Bytes	Description	Length
1	INS byte of the following command sent to the UC (Coded according to TS 101 206-3 [2])	1 byte
2	P1 of the following command sent to the UC (coded according to TS 101 206-3 [2])	1 byte
3	P2 of the following command sent to the UC (coded according to TS 101 206-3 [2])	1 byte
4	L_c of the following command sent to the UC (coded according to TS 101 206-3 [2])	1 byte
5 - (4+X)	The data field of the following command sent to the UC (Coded according to TS 101 206-3 [2])	X bytes

c/ key number (indicated in the command header if SM is an IC card).

Output: a/ MAC (UC command header, data):

Table 8: Cryptogram

Bytes	Description	Length
1 - X	MAC	X bytes

b/ status.

NOTE: The length x depends on the cryptographic algorithm application. In case of TESA 7, x = 8.

For the following functions in the UC, the COMPUTE MAC function is applicable when AC = PRO:

CREATE FILE
DELETE FILE
EXTEND
EXECUTE
UPDATE BINARY
UPDATE RECORD
CREATE RECORD
INVALIDATE
REHABILITATE
WRITE BINARY
WRITE RECORD
DECREASE
CREATE RECORD

The purpose of these functions is to prove that the SM has been involved in a given transaction and that the function it was to perform has been accomplished.

8.2.3 COMPUTE CRYPTOGRAM

This function shall be used to compute the cryptogram used in the user card in the EXTERNAL AUTHENTICATION command.

Pre-conditions: the challenge used shall be obtained from the UC by an ASK RANDOM function and transmitted by the terminal to the SM using a GIVE RANDOM;

the keyset shall be selected using the SELECT KEYSET function. The relevant key file shall be the one containing the keys after diversification

Input: a/ command header/trailer (COMPUTE CRYPTOGRAM);

b/ key number (indicated in the command header if SM is an IC card).

Output: a/ cryptogram;

Table 9: Cryptogram

Bytes	Description	Length
1 - X	MAC	X bytes

b/ status.

8.2.4 COMPUTE MAC External World (EW)

The purpose of this function is to add a cryptogram to the input data given to the SM to provide the external world with a proof of

- the authenticity of the SM involved in the transaction;
- the integrity of the data during transmission from SM to the external world.

The input data for this function contains parameters of the same structure as a command header in the UC.

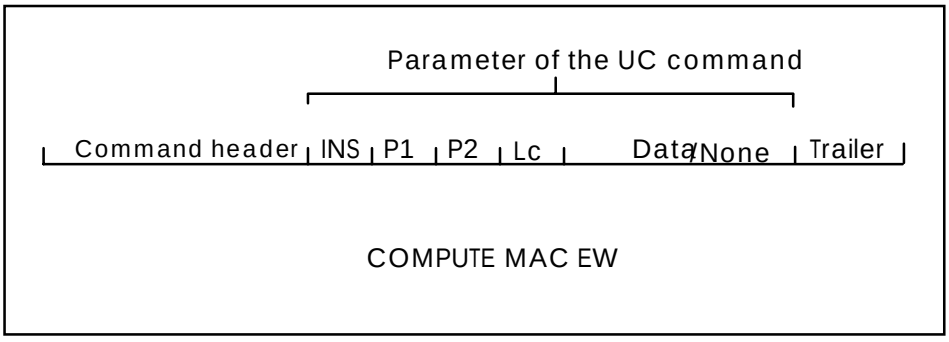

Figure 10: General structure of COMPUTE MAC EW

Pre-conditions: the challenge used shall be obtained from the SM by an ASK PARAMETER function before.

Input: a/ command header/trailer (COMPUTE MAC EW);

b/ data:

Table 10: Data

Bytes	Description	Length
1	INS byte of the following command sent to the UC	1 byte
2	P1 of the following command sent to the UC	1 byte
3	P2 of the following command sent to the UC	1 byte
4	L_c of the following command sent to the UC	1 byte
5 - (4+X)	The data field of the following command sent to the UC	X bytes

c/ key number (indicated in the command header if SM is an IC card).

Output: a/ cryptogram (command header, data):

Table 11: Cryptogram

Bytes	Description	Length
1 - X	MAC	X bytes

b/ status.

NOTE: The length x depends on the cryptographic algorithm application. In case of TESA 7, x = 8.

8.2.5 DECREASE (SM)

The purpose of this function is to decrease a balance in a SM by the amount that will be added to the counter value of a valid UC (reload). The SM shall check if the indicated key is allowed to perform this function.

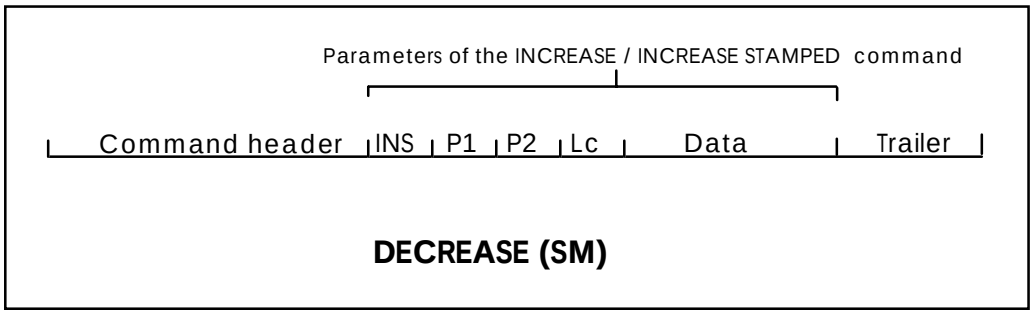

Figure 11: Structure of DECREASE (SM)

Pre-conditions: the challenge used shall be obtained from the UC using an ASK RANDOM function and transmitted to the SM using a GIVE RANDOM;

the keyset shall be selected using the SELECT KEYSET function. The relevant key file shall be the one containing the keys after diversification;

the file to be decreased in the SM has to be selected.

NOTE: It is recommended that the remaining value in the UC is obtained to check if the reload amount will exceed the maximum value defined before decreasing the SM. This could be done by issuing a DECREASE function with the amount equal to zero to the UC.

Input: a/ command header/trailer (DECREASE (SM));

b/ Key number (included in the command header);

c/ data (parameters of the INCREASE or INCREASE STAMPED (UC) command).

Table 12: Data field of the DECREASE (SM) command

Bytes	Description	Length
1	INS byte of the following INCREASE or INCREASE STAMPED command sent to the UC (coded according to TS 101 206-3 [2])	1 byte
2	P1 of the following INCREASE or INCREASE STAMPED command sent to the UC (coded according to TS 101 206-3 [2])	1 byte
3	P2 of the following INCREASE or INCREASE STAMPED command sent to the UC (coded according to TS 101 206-3 [2])	1 byte
4	The L_c of the following INCREASE or INCREASE STAMPED command (coded according to TS 101 206-3 [2])	1 byte
5 - (4+X)	The data field of the following INCREASE or INCREASE STAMPED command sent to the UC (coded according to TS 101 206-3 [2])	X bytes

Output: a/ MAC:

Table 13: DECREASE (SM) response

Bytes	Description	Length
1 - X	MAC	X bytes

NOTE: The length x depends on the cryptographic algorithm application. In case of TESA 7, x = 8.

b/ status.

8.3 Functions verifying cryptograms or MAC

If TESA 7 is used, a cryptogram is computed according to TESA 7 mode "authentication" (see EN 726-2 [1], subclauses A3.1 and A3.2) and MAC is computed according to TESA 7 mode "MAC" (see EN 726-2 [1], subclauses A2.3 and A2.4). In other parts of the present document the result of the MAC computation may be also referred as "cryptogramm".

8.3.1 VERIFY MAC

The purpose of this function is to verify a MAC performed in the UC. The SM shall check if the indicated key is allowed to perform this function.

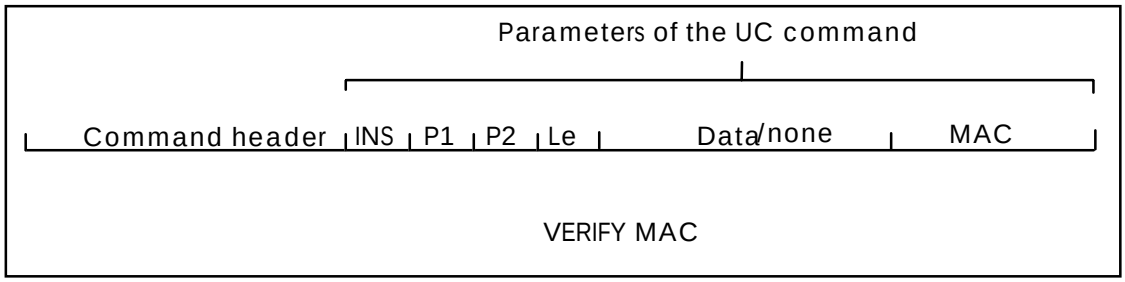

Figure 12: Structure of VERIFY MAC

Pre-conditions: the challenge used shall be obtained from the SM using an ASK PARAMETER function and transmitted to the UC using a GIVE RANDOM;

the keyset shall be selected using the SELECT KEYSET function. The relevant key file shall be the one containing the keys after diversification.

Post-conditions: the challenge used by the command shall be kept and usable for the next command with pre-condition ASK PARAMETER;

Input: a/ command header/trailer (VERIFY MAC);

b/ Key number (indicated in the command header);

c/ data (parameters of the previous UC command/response).

Table 14: Data

Bytes	Description	Length
1	INS byte of the previous command sent to the UC (Coded according to TS 101 206-3 [2])	1 byte
2	P1 of the previous command sent to the UC (coded according to TS 101 206-3 [2])	1 byte
3	P2 of the previous command sent to the UC (coded according to TS 101 206-3 [2])	1 byte
4	The L_e of the previous command sent to the UC (Coded according to TS 101 206-3 [2])	1 byte
5 - (4+X)	The data field of the previous response sent by the UC (Coded according to TS 101 206-3 [2])	X bytes

Output: a/ data: none;

b/ status.

For the following functions in the UC which provide MACs, the VERIFY MAC function is applicable:

- READ BINARY STAMPED;
- READ RECORD STAMPED.

The cryptogram shall be in accordance with TS 101 206-3 [2]. The key indicated in the command header of the VERIFY MAC function, shall be the same as the one used to compute the MAC in the UC.

8.3.2 UPDATE (SM)

The purpose of this function is to update information in the SM. Only after verification of a MAC performed in the UC with a READ RECORD STAMPED, shall the information be updated. The SM shall check if the indicated key is allowed to perform this function.

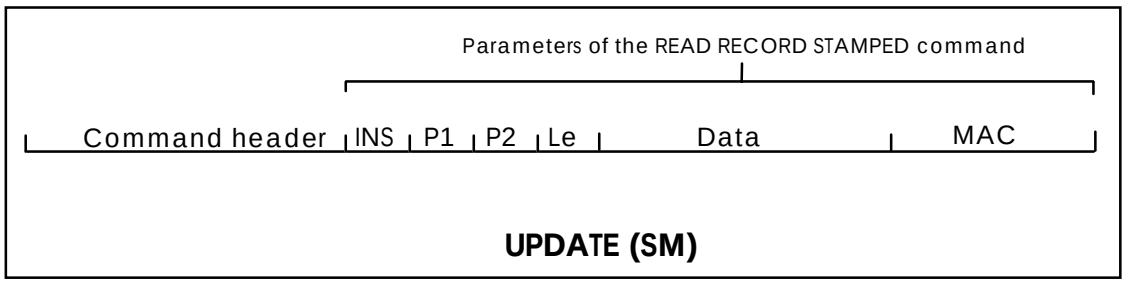

Figure 13: Structure of UPDATE (SM)

Pre-conditions: the challenge used shall be obtained from the SM using an ASK PARAMETER function and transmitted to the UC using a GIVE RANDOM;

the keyset shall be selected using the SELECT KEYSET function. The relevant key file shall be the one containing the keys after diversification;

the file to be updated in the SM shall be selected.

Input: a/ command header/trailer(UPDATE (SM));

b/ Key number (included in the command header);

c/ data (parameters of the previous READ RECORD STAMPED (UC) command/response).

Table 15: Data field of the UPDATE (SM) command

Bytes	Description	Length
1	INS byte of the previous READ RECORD STAMPED command sent to the UC (Coded according to TS 101 206-3 [2])	1 byte
2	P1 of the previous READ RECORD STAMPED command sent to the UC (coded according to TS 101 206-3 [2])	1 byte
3	P2 of the previous READ RECORD STAMPED command sent to the UC (coded according to TS 101 206-3 [2])	1 byte
4	The L_e of the previous READ RECORD STAMPED command sent to the UC (Coded according to TS 101 206-3 [2])	1 byte
5 - (4+X)	The data field of the previous READ RECORD STAMPED response sent by the UC (Coded according to TS 101 206-3 [2])	X bytes

Output: a/ data: none;

b/ status.

The MAC shall be in accordance with TS 101 206-3 [2]. The key indicated in the command header of the UPDATE (SM) function, shall be the same as the one used to compute the MAC by the READ RECORD STAMPED function in the UC.

8.3.3 INCREASE (SM)

The purpose of this function is to increase a balance in the SM by exactly the same amount that was decreased before from the counter value in a valid UC in the same transaction. The SM shall check if the indicated key is allowed to perform this function.

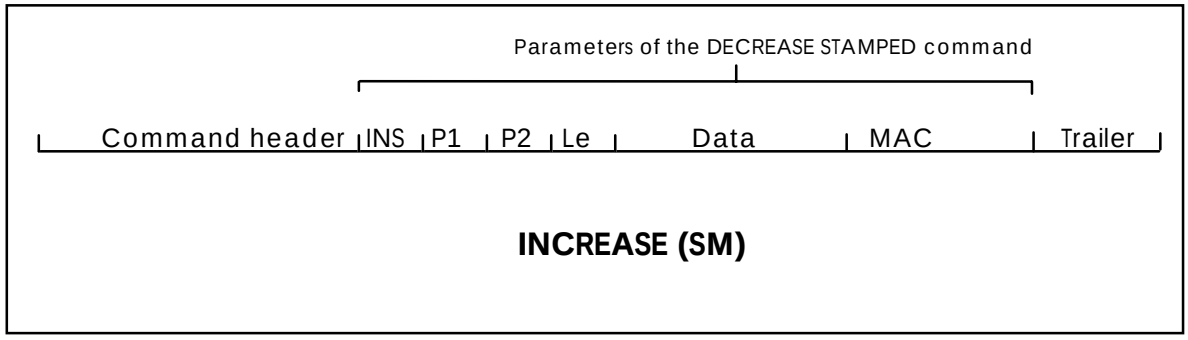

Figure 14: Structure of INCREASE (SM)

Pre-conditions: the challenge used shall be obtained from the SM using an ASK PARAMETER function and transmitted to the UC using a GIVE RANDOM;

the keyset shall be selected using the SELECT KEYSET function. The relevant key file shall be the one containing the keys after diversification;

the file to be increased in the SM shall be selected;

this function follows a DECREASE STAMPED of the UC and the data (value deducted) decreased in the UC shall be used as data (value added) increased in the SM.

Input: a/ command header/trailer (INCREASE (SM));

b/ Key number (included in the command header);

c/ data (parameters of the previous DECREASE STAMPED (UC) command/response).

Table 16: Data field of the INCREASE (SM) command

Bytes	Description	Length
1	INS byte of the previous DECREASED STAMPED sent to the UC (Coded according to TS 101 206-3 [2])	1 byte
2	P1 of the previous DECREASED STAMPED sent to the UC (coded according to TS 101 206-3 [2])	1 byte
3	P2 of the previous DECREASED STAMPED sent to the UC (coded according to TS 101 206-3 [2])	1 byte
4	The L_e of the previous DECREASED STAMPED sent to the UC (Coded according to TS 101 206-3 [2])	1 byte
5 - (4+X)	The data field of the previous DECREASE STAMPED response sent by the UC (Coded according to TS 101 206-3 [2])	X bytes

Output: a/ data: none;

b/ status.

The MAC shall be in accordance with TS 101 206-3 [2]. The key indicated in command header of the INCREASE (SM) function, shall be the same as the one used to compute the MAC by the DECREASE STAMPED function in the UC.

8.3.4 VERIFY CRYPTOGRAM

The purpose of this function is to authenticate an application in the user card by verifying the cryptogram performed in the UC after an INTERNAL AUTHENTICATION (INT.AUTH) command.

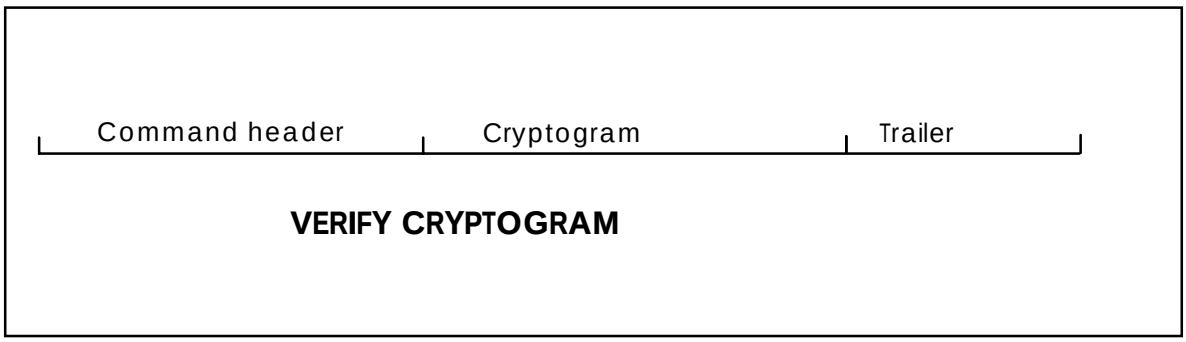

Figure 15: Structure of VERIFY CRYPTOGRAM

Pre-conditions: the challenge used shall be generated by the SM after an ASK PARAMETER command and transmitted to the UC using an INTERNAL AUTHENTICATION command;

the keyset shall be selected using the SELECT KEYSET function. The relevant key file shall be the one containing the keys after diversification.

Input: a/ command header/trailer (VERIFY CRYPTOGRAM);

b/ key number (indicated in the command header if SM is an IC card).

c/ cryptogram.

Table 17: Data field of the VERIFY CRYPTOGRAM command

Bytes	Description	Length
1 - x	Crytpogram	X bytes

Output: a/ data: none;

b/ status.

The cryptogram shall be in accordance with TS 101 206-3 [2]. The key indicated in the command header of the VERIFY CRYPTOGRAM function, shall be the same as the one used to compute the cryptogram in the UC.

8.4 Functions for downloading of keys from the SM to the UC

The functionality for downloading of keys from the SM to the UC for a SM being an IC card is described in annex A, clause A.7.

8.5 Functionality on SM - EW interface

If the SM is an IC card, the functions and the set of commands according to TS 101 206-3 [2] shall be used to handle the management of SM files belonging to telecom applications specified by ETSI.

In any other case, the functionality of the SM - EW interface is out of the scope of the present document.

NOTE: It is strongly recommended that the integrity of functions and commands is maintained.

8.6 Linking functions with keys

A linking of functions to key numbers within the SM is needed which limits the use of any function to only those key numbers which this function is allowed to use.

The SM shall handle a set of information in order to link each key to its respective functions:

- relevant key qualifier;

Table 18: Key qualifier

File level	Relevant key qualifier
MF	IC card manufacturing references, card personalizer ID, key file version, EF_{KEY} type
DF (application level)	AID, key file version, EF_{KEY} type
DF (under application level)	Proprietary part of AID, key file version, EF_{KEY} type

- master key;
- algorithm ID;
- algorithm;
- diversified key;
- functions allowed to use this key;
- access conditions to change this set of information.

Details of how this linking is managed by the SM is out of the scope of the present document.

NOTE: A SM contains all secrets to compute cryptograms normally coming from a UC. Theoretically a fraudulent terminal can bypass the presence of a valid UC by asking the SM to compute all cryptograms as coming from a UC and presenting them to the SM for verification.

8.7 Limitations of the usage of keys

The usage of keys for the following functions shall be restricted by the SM:

- COMPUTE MAC (EXECUTE);
- COMPUTE MAC (DECREASE);
- VERIFY MAC (READ BINARY STAMPED);
- VERIFY MAC (READ RECORD STAMPED);
- VERIFY CRYPTOGRAM (INTERNAL AUTHENTICATION);
- UPDATE (SM);
- INCREASE (SM);
- DECREASE (SM);
- COMPUTE MAC EW.

A more detailed description of the functionality in case of a SM being an IC card is given in annex A.

9 Data elements

Application independent data elements and those data elements which are necessary to support EN 726-5 [3] are described in the following subclauses.

9.1 Data for SM identification

9.1.1 SM identifier

Purpose: data element to uniquely identify the SM (device).

Status: mandatory for auto billing application, see EN 726-5 [3].

Format: 4 bytes.

Contents: register, maintained by ISO/IEC according to ISO/IEC 10202-4 [5].

9.2 UC data

This is an informative listing of related data elements in the UC and on its interface and does not define requirements for the storage or handling of this data in the SM.

9.2.1 IC serial number of user card

Purpose: data element to uniquely identify the IC within each UC.

Status: mandatory.

Format: 4 byte binary.

Contents: at the discretion of the IC manufacturer.
stored in the EF_{ICC} of the UC, see TS 101 206-3 [2].

9.2.2 Application Expiry date

Purpose: data element to define on which date the application ceases to be valid for use.

Status: mandatory.

Format: 3 bytes Binary Coded Decimal (BCD) in the format YYMMDD.

Contents: for the prepayment application the expiry date is stored in EF_{PAY} of the UC, see EN 726-5 [3];
for the auto billing application the expiry date is stored in EF_{AUTO} of the UC, see EN 726-5 [3].

9.2.3 Application Identifier (AID)

Purpose: data element to uniquely identify the application and the relevant keyset.

Status: optional, but strongly recommended.

Format: 1-16 bytes binary (see ISO/IEC 7816-5 [4]).

Contents: according to ISO/IEC 7816-5 [4].

9.2.4 Key file version

Purpose: data element to uniquely identify the version of a key file within an application.

Status: mandatory.

Format: 1 byte binary coded.

Contents: keyset version number, stored in a UC in $EF_{KEY\_OP/\_MAN}$ according to TS 101 206-3 [2], in a SM being an IC card in $EF_{KEY\_OP/\_MAN}$ according to the present document.

9.2.5 Algorithm identifier

Purpose: data element linked to any key to uniquely identify the algorithm and its parameters to perform a cryptographic calculation.

Status: mandatory.

Format: 1 byte binary, stored in a UC in $EF_{KEY\_OP/\_MAN}$ according to EN 726-3 [3], subclauses 10.6 and 10.7, in SM being an IC card in $EF_{KEY\_OP/\_MAN}$ according to the present document.

Contents: according to a register to be established and maintained.

9.2.6 Amount

Purpose: data element containing the amount deducted.

Status: mandatory for prepayment application:
Deducted from EF_{AMOUNT} of UC and SM, see EN 726-5 [3] subclause 5.4.3.

Format: 3 bytes binary coded.

Contents: amount to be deducted.

9.2.7 Value

Purpose: data element containing the new counter value of the specific application.

Status: mandatory for prepayment application:
Deducted from EF_{AMOUNT} of UC and SM, see EN 726-5 [3].

Format: 1 + X bytes binary coded.

Contents: length of the counter value, counter value.

9.2.8 Type of units

Purpose: data element to indicate the kind of units and the equivalent of one unit of the counter of EF_{AMOUNT} UC.

Status: mandatory for prepayment application:
Stored in EF_{PAY} of UC, see EN 726-5 [3].

Format: 4 bytes BCD.

Contents: Telecom unit, equivalence of one unit.

Coding: Telecom unit 3 bytes according to ISO 4217 [6] and EN 726-5 [3].

9.3 Status conditions returned by the SM

For consistency of an application using different types of SM the responses to different functions shall have the same meaning.

9.3.1 Functions versus possible status responses

In this subclause only responses concerning the additional functions for the SM are defined.

For functions used in a SM described in TS 101 206-3 [2], refer to relevant responses there.

Tables 16 and 17 show for each supplementary function the possible status conditions returned (marked by *).

Table 19: Status responses

	Security Status									Memory Status			
Functions	98 AD	98 02	98 04	98 08	98 10	98 35	98 40	98 50		92 0X	92 10	92 20	92 40
SELECT KEYSET	*	-	-	-	-	-	-	-		-	-	-	*
DIVERSIFY KEYS	*	-	-	-	*	-	-	-		*	-	-	*
ASK PARAMETER	*	-	-	-	-	-	-	-		-	-	-	*
COMPUTE CRYPTOGRAM	*	*	-	-	*	*	-	-		-	-	-	*
VERIFY CRYPTOGRAM	*	*	*	-	*	*	-	-		-	-	-	*
COMPUTE MAC	*	*	-	-	*	*	-	-		-	-	-	*
VERIFY MAC	*	*	*	-	*	*	-	-		-	-	-	*
UPDATE (SM)	*	*	*	-	*	*	-	-		*	-	-	*
INCREASE (SM)	*	*	*	-	*	*	-	-		*	-	-	*
DECREASE (SM)	*	*	-	-	*	*	-	-		*	-	-	*
COMPUTE LOAD KEY	*	*	-	-	*	*	-	-		-	-	-	*
COMPUTE MAC EW	*	*	-	-	*	*	-	-		-	-	-	*

The status condition "Command out of sequence" ('98 AD') returned by the SM can appear after any command depending on the application and on the implementation of the sequence control.

Table 20: Status response

	Reference Status					Application Independent Errors						OK	
Functions	94 00	94 02	94 04	94 08		6E XX	6D XX	6F XX	6B XX	67 XX		90 00	9F XX
SELECT KEYSET	-	-	*	-		*	*	*	**	*		*	-
DIVERSIFY KEYS	*	-	-	*		*	*	*	**	*		*	-
ASK PARAMETER	-	*	-	-		*	*	*	**	*		*	-
COMPUTE CRYPTOGRAM	-	-	-	-		*	*	*	**	*		*	*
VERIFY CRYPTOGRAM	-	-	-	-		*	*	*	**	*		*	-
COMPUTE MAC	-	-	-	-		*	*	*	*	*		*	*
VERIFY MAC	-	-	-	-		*	*	*	*	*		*	-
UPDATE (SM)	*	-	-	*		*	*	*		*		*	-
INCREASE (SM)	*	-	-	*		*	*	*		*		*	*
DECREASE (SM)	*	-	-	*		*	*	*		*		*	*
COMPUTE LOAD KEY	-	-	-	-		*	*	*		*		*	*
COMPUTE MAC EW	-	-	-	-		*	*	*		*		*	*

9.3.1.1 Security management

Table 21: Security management

SW1	SW2	Error description
'98'	'02'	- No Card Holder Verification information (CHV) and/or key defined - There were no AC to be fulfilled for this DF
'98'	'04'	- AC not fulfilled - Wrong cryptogram verification - Unsuccessful CHV verification but verify CHV mechanism still possible (number of false consecutive verifications < N) - Unsuccessful UNBLOCK CHV verification but verify UNBLOCK CHV mechanism still possible (number of false consecutive verifications < 10)
'98'	'08'	- In contradiction with CHV status - The CHV protecting the file at the CP is not blocked
'98'	'10'	- In contradiction with the invalidation status
'98'	'35'	- No ASK RANDOM/GIVE RANDOM before
'98'	'40'	- Unsuccessful CHV verification, verify CHV mechanism no longer possible (number of false consecutive verifications ≥ N) - Unsuccessful UNBLOCK CHV verification, verify UNBLOCK CHV mechanism no longer possible (number of false consecutive verifications ≥ 10)
'98'	'50'	- Increase/Decrease cannot be performed Maximum/minimum value reached)
'98'	'AD'	- Command out of sequence

9.3.1.2 Memory management

Table 22: Memory management

SW1	SW2	Error description
'92'	'0X'	- Update successful but after using an internal retry routine x times
'92'	'10'	- Insufficient memory space available
'92'	'20'	- File ID is already existing in this parent file (MF, DF)
'92'	'40'	- Memory problem

9.3.1.3 Referencing management

Table 23: Referencing management

SW1	SW2	Error management
'94'	'00'	- No EF selected as current - EF not selected
'94'	'02'	- Out of range (invalid address)
'94'	'04'	- File ID not found - Record not found (see note) - Pattern not found
'94'	'08'	- Current file is inconsistent with the command

NOTE: The use of 94 04 for record not found is not recommended for future applications, 94 02 should be used instead.

9.3.1.4 Application independent errors

Table 24: Application independent errors

SW1	SW2	Error description
'6E'	'XX'	- Wrong instruction class given in the command
'6D'	'XX'	- Unknown instruction code given in the command
'6F'	'XX'	- Technical problem with no diagnostic given (command aborted)
'6B'	'XX'	- Incorrect parameters P1 or P2
'67'	'XX'	- Incorrect parameter P3

9.3.1.5 Responses to commands which are correctly executed or supporting chaining mechanism

Table 25: Responses to commands which are correctly executed or supporting chaining mechanism

SW1	SW2	Error description
'90'	'00'	- Normal ending (ACK) of the command
'9F'	'XX'	- Length 'XX' of the response data

Annex A (normative): The case of a SM in the form of an IC card

A.1 Purpose

This annex describes the SM in the form of an IC card for a system which handles user cards in accordance with TS 101 206-3 [2]. The following limitations are given:

a) the SM handles one and only one terminal (the SM handles derived keys based on symmetric algorithms only);

b) the SM supports the minimum subset of functionality described in the main part of the present document.

Other configurations are out of the scope of this annex.

A.2 General requirements of a SM in the form of an IC card

Physical characteristics and electronic signals and transmission protocols shall be in accordance with TS 101 206-3 [2].

A.3 Card architecture

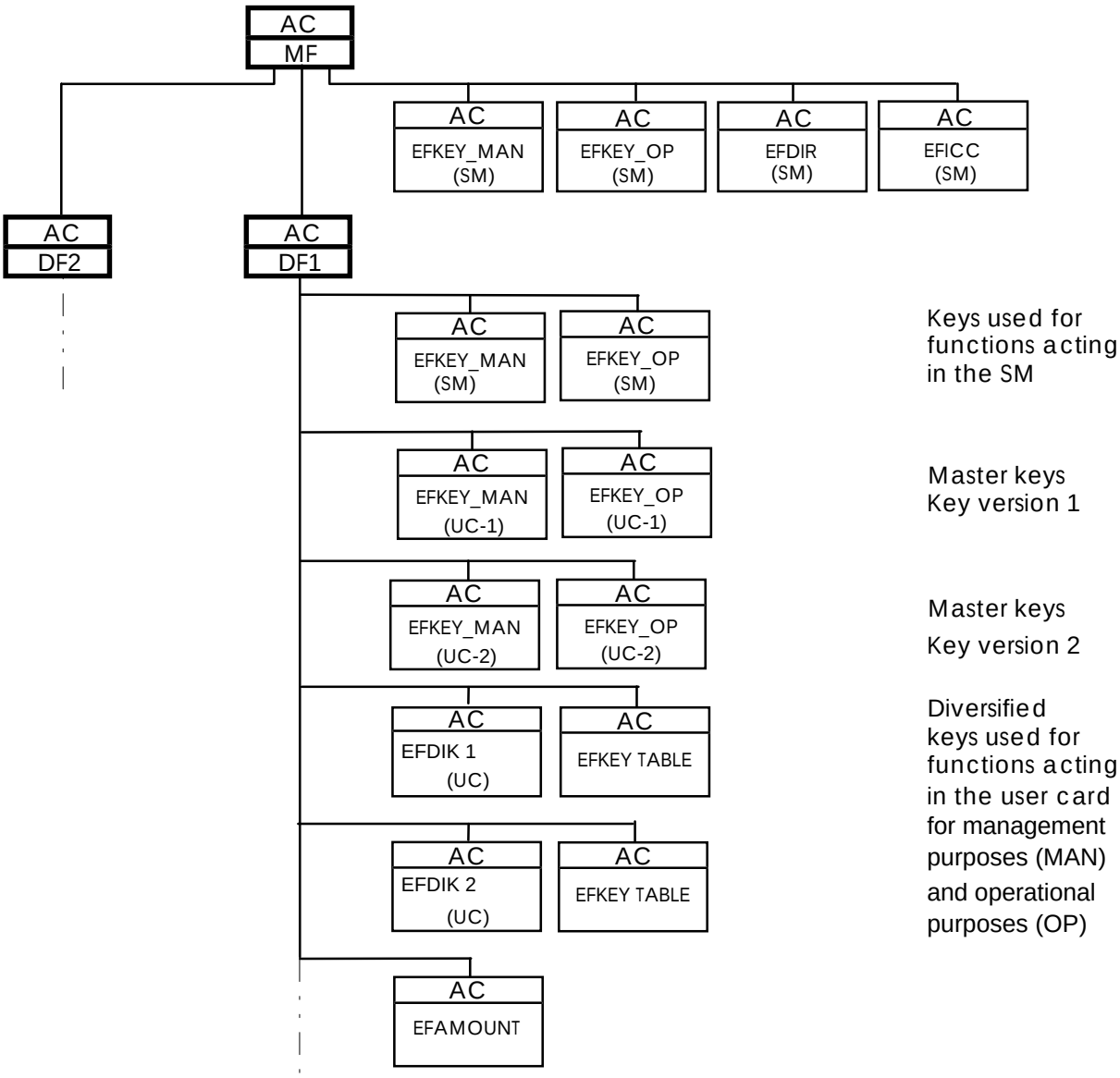

Figure A.1: An example of key architecture for a SM (IC card)

For functions performed on the SM by the management system, fixed keys are used to fulfil AC as described in TS 101 206-3 [2]. For functions to be performed on the UC, the SM uses diversified keys as described in the present document.

A.4 Description of files

A.4.1 Master file

Status: mandatory.

Reference: see ISO/IEC 7816-4 [4].

A.4.2 EF_{ICC}

Status: mandatory.

Contains a SM identifier.

Reference: see TS 101 206-3 [2], subclause 10.4.

A.4.3 EF_{DIR}

Status: optional, but recommended.

Reference: see TS 101 206-3 [2] subclause 6.3.

A.4.4 EF_{KEY_MAN} (SM)

Purpose: see TS 101 206-3 [2] .

Status: mandatory.

Reference: see TS 101 206-3 [2] subclause 10.6.

A.4.5 EF_{KEY_OP} (SM)

Purpose: see TS 101 206-3 [2].

Status: optional.

Reference: see TS 101 206-3 [2] subclause 10.7.

A.4.6 DFx

Status: at least one DF mandatory.

Reference: see TS 101 206-3 [2] subclause 9.2.3.

A.4.7 Files containing master keys: $EF_{KEY\_MAN/OP}$ (UC)X

These key files contain the master keys corresponding to the keyset version X in the UC.

Table A.1: EF_{KEY_MAN} (UC)

File ID: '20 XX' EF_{KEY_MAN} (UC) (note)		Optional
Type of file: '00' (Transparent EF)		
AC: LOAD KEY FILEUPDATE INVALIDATE REHABILITATE	PROPRO NEV NEV	

Bytes	**Description**	**Length**
1	Keyset version	1 byte
2	Length of key 0 (X bytes)	1 byte
3	Algorithm ID for key 0	1 byte
4 - 3 + X	Key 0	X bytes
4 + X	Length of key 1	1 byte
etc.		

NOTE: Key file number is indicated by 'XX'.

Table A.2: EF_{KEY_OP} (UC)

File ID: '21 YY' EF_{KEY_OP} (UC) (note 1)		Mandatory
Type of file: '00' (Transparent EF)		
AC: LOAD KEY FILEUPDATE INVALIDATE REHABILITATE	 PRO (note 2)PRO (note 2) PRO (note 2) PRO (note 2)	
Bytes	**Description**	**Length**
1	Keyset version	1 byte
2	Length of key 0 (X bytes)	1 byte
3	Algorithm ID for key 0	1 byte
4 - 3 + X	Key 0	X bytes
4 + X	Length of key 1	1 byte
etc.		
NOTE 1: Key file number is indicated by 'YY'. NOTE 2: It is up to the SM provider to link the function to any of the keys of the relevant EF_{KEY_MAN} (SM).		

End of file is indicated by the key length being 0.

The structure of this file has to be the mirror of the relevant EF_{KEY} file of the UC, even if some keys are not used in SM.

For those master keys which are not available in the SM, the coding shall be according to table A.3.

Table A.3: Coding of master keys which are not available (EF_{KEY_MAN} (UC) or EF_{KEY_OP} (UC))

Bytes	**Description**	**Length**
....	.....	.....
n	Length of key 1 ('XX' bytes)	1 byte
(n+1)	Algorithm ID for key 1 ('FF', see note)	1 byte
(n+2) -(n+1+ XX)	Key X	XX bytes
.....	.....	.....
NOTE: Algorithm ID = 'FF' indicates that the content is not a valid master key and use of this master key shall never be allowed.		

A.4.8 File for linking keys to functions: EFkeytable

This file contains a table linking the keys of EF_{KEY_OP} (UC) or EF_{KEY_MAN} (UC) to functions and is the responsibility of the SM provider or authority delegated by him. For each EF_{DIK} (OP or MAN) one EFkeytable shall exist.
Only the relevant key may be used for those functions (see subclause 8.7) which are linked in the EFkeytable.

Table A.4: EFkeytable

File ID: '02 XX' EFkeytable (note)		Mandatory
Type of file: '01' (EF with a linear fixed structure)		
AC: READ NEV CREATE...EXECUTE PRO UPDATE NEV WRITE NEV INVALIDATE NEV REHABILITATE NEV		
Bytes	**Description**	**Length**
1	INS of the SM command	1byte
2	INS of the UC command which is included in the cryptogram	1 byte
3	Key number linked to the SM command	1 byte
4 - 5	File ID of the relevant file in the SM	2 bytes
6	INS of the SM command	1 byte
7	INS of the UC command which is included in the cryptogram	1 byte
8	Key number linked to the SM command	1 byte
9 - 10	File ID of the relevant file in the SM	2 bytes
.....	.....	.....
NOTE: 'XX' = '00' means EFkeytable (MAN). 'XX' = '01' means EFkeytable (OP).		

A.4.9 EF_{AMOUNT} (SM)

This file contains the counter value which may only be changed by the INCREASE (SM) or by DECREASE (SM) functions.

Table: A.5: EF_{AMOUNT} (SM)

File ID: '12 XX' EF_{AMOUNT} (SM) (note 1)		Optional
Type of file: '03' (Cyclic EF)		
AC:		
READ	PRO (note 2)	
CREATE...EXECUTE	PRO (note 2)	
DECREASE	PRO (notes 2 and 3)	
INCREASE	PRO (notes 2 and 3)	
INVALIDATE	PRO (note 2)	
REHABILITATE	PRO (note 3)	

Bytes	Description	Length
1 - 3	Counter value	3 bytes

NOTE 1: 'XX' is equal to '00' except when there is more than one EF_{AMOUNT} (SM) under the same DF.

NOTE 2: It is up to the security module provider to link the function to any of the keys of the relevant EF_{KEY_MAN}SM.

NOTE 3: ACs INCREASE/DECREASE replace the ACs UPDATE/WRITE like in a UC, as described in TS 101 206-3 [2].

Bytes 1 - 3 are represented in HEX notation.

See also EN 726-5 [3], subclause 5.4.3.

A.4.10 Files containing diversified keys: EF_{DIK_OP} (UC) and $EF_{DIK\ MAN}$ (UC)

These Keyfiles contain the diversified keys derived from the master keys stored in EF_{KEY_OP} (UC) / EF_{KEY_MAN} (UC) of the SM corresponding to the keyset version x in the UC.

Table A.6: EF_{DIK_MAN} (UC)

File ID: '10 00' EF_{DIK_MAN} (UC)		Optional
Type of file: '00' (Transparent EF)		
AC:		
LOAD KEY FILE		
UPDATE	NEVNEV (note 2)	
INVALIDATE	NEV	
REHABILITATE	NEV	
Bytes	**Description**	**Length**
1	Keyset version	1 byte
2	Length of key 0 (X bytes)	1 byte
3	Algorithm ID for key 0	1 byte
4 - 3 + X	Key 0	X bytes
4 + X	Length of key 1	1 byte
....	.....	.....

NOTE 1: It is up to the SM provider to link the function to any of the keys of the relevant EF_{KEY_MAN} (SM).

NOTE 2: The only function possible after the creation of this file shall be an internal update using the function DIVERSIFY KEYSET.

Table A.7: EF_{DIK_OP} (UC)

File ID: '11 00' EF_{DIK_OP} (UC)		Mandatory
Type of file: '00' (Transparent EF)		
AC:		
UPDATE	NEV (note 2)	
LOAD KEY FILE	NEV	
INVALIDATE		
REHABILITATE	NEV	
	NEV	
Bytes	**Description**	**Length**
1	Keyset version	1 byte
2	Length of key 0 (X bytes)	1 byte
3	Algorithm ID for key 0	1 byte
4 - 3 + X	Key 0	X bytes
4 + X	Length of key 1	1 byte
....	.....	.....

NOTE 1: It is up to the SM provider to link the function to any of the keys of the relevant EF_{KEY_MAN}SM.

NOTE 2: The only function possible after the creation of this file shall be an internal update using the function DIVERSIFY KEYSET.

If length of key is '00' this indicates end of file.

If length of key is '01' this indicates that this key is not used. The following byte indicates length of next key.

Even if master keys are not present in this SM, all keys in the UC shall be included in the table up to the highest key number stored in the UC.

For those master keys which are not available the coding shall be according to table A.3.

A.5 Commands

A.5.1 General remarks

If the SM is an IC card, the commands described in TS 101 206-3 [2] clause 8 are acting on MF, DFs and EFs. Additional commands are described in the following subclauses.

In the case the SM is an IC card the command header consists of the CLA; INS, P1, P2 and L_c elements and the trailer consists of the L_e element, they are coded as specified hereafter for the commands involved.

A.5.2 Additional commands

Table A.8: Coding of commands

Command	INS	P1	P2
SELECT KEYSET	'50'	'00'	'00'
DIVERSIFY KEYSET	'52'	EF_{DIK} number	'00'
ASK PARAMETER	'54'	Type	'00'
COMPUTE LOAD KEY	'70'	EF_{DIK} number	Key number
COMPUTE CRYPTOGRAM	56'	'00'	Key number
VERIFY CRYPTOGRAM	'58'	'00'	Key number
COMPUTE MAC	'8A'	'00'	Key number
VERIFY MAC	'8E'	'00'	Key number
UPDATE (SM)	'5A'	'00'	Key number
INCREASE (SM)	'5C'	'00'	Key number
DECREASE (SM)	'5E'	'00'	Key number

A.5.2.1 SELECT KEYSET

Table A.9: Coding of the SELECT KEYSET command

CLA	As defined in TS 101 206-3 [2] subclause 9.2 .
INS	'50
P1	'00'
P2	'00'
L_c field	Length of the data field.
Data field	Key qualifier.
L_e field	Empty.

Table A.10: Coding of the data field of the SELECT KEYSET command (in case of a MF)

Bytes	Description	Length
1 - 4	IC card manufacturing references (coded according to TS 101 206-3 [2])	4 bytes
5	Card personalizer ID (coded according to TS 101 206-3 [2])	1 byte
6 - 7	File ID of the EF_{KEY}.	2 bytes
8	Keyfile version (coded according to TS 101 206-3 [2])	1 byte

Table A.11: Coding of the data field of the SELECT KEYSET command (in case of a DF)

Bytes	Description	Length
1 - X	AID (coded according to TS 101 206-3 [2])	1 - 16 bytes
(X+1) - (X+2)	File ID of the EF_{KEY}	2 bytes
X+3	Keyfile version (coded according to TS 101 206-3 [2])	1 byte

A.5.2.2 DIVERSIFY KEYSET

Table A.12: Coding of the DIVERSIFY KEYSET command

CLA	As defined in TS 101 206-3 [2] subclause 9.2.
INS	'52'
P1	EF_{DIK} number.
P2	'00'
L_c field	Length of the data field.
Data field	Diversification data.
L_e field	Empty.

Coding of EF_{DIK} number (P1):

a) '00'EF_{DIK} number 1: used to temporarily store relevant keys, diversified from selected master keyset, to perform the following function.

b) '01'EF_{DIK} number 2: used to temporarily store the diversified keyset to be down loaded.

Table A.13: Coding of the data field of the DIVERSIFY KEYSET command

Bytes	Description	Length
1	Algorithm ID for diversification	1 byte
2 - X	Diversification data	1 - 16 bytes
NOTE: This command includes the functionality to provide the contents of the EF_{DIK} (keyfile version, key length, algorithm identifier and diversified keys).		

For coding of algorithm ID see TS 101 206-3 [2] subclause 7.6.5

A.5.3 Commands handling parameters for cryptography

A.5.3.1 ASK PARAMETER

Table A.14: Coding of the ASK PARAMETER command

CLA	As defined in TS 101 206-3 [2] subclause 9.2.
INS	'54'
P1	Type.
P2	'00'
L_c field	Empty.
Data field	Empty.
L_e field	Maximum length of data expected in response.

Coding of types (P1):

a) '00'challenge (random number);

b) '01'challenge (counter related to the keyset containing masterkeys selected before.

NOTE: "Ask Parameter" triggers the modification of the counter.

In case there is no counter linked to the selected keyset, back tracking mechanism to the next existing counter at higher level shall be used.

Table A.15: Coding of the ASK PARAMETER response

Bytes	Description	Length
1 - X	Challenge	X bytes
NOTE: If the algorithm TESA 7 is used, the length of the challenge is fixed to 8 bytes.		

A.5.4 Commands to compute a cryptogram or MAC

Mechanism to create and link counters have to be offered by the operating system.

Implementation is out of scope of the present document.

A.5.4.1 COMPUTE LOAD KEY

Table A.16: Coding of the COMPUTE LOAD KEY command

CLA	As defined in TS 101 206-3 [2] subclause 9.2.
INS	'70'
P1	EF_{DIK} number.
P2	Key number.
L_c field	Length of the data field.
Data field	Data sent to the SM.
L_e field	Maximum length of data expected in response.

Coding of EF_{DIK} number (P1):

'00'EF_{DIK} number 1; (containing div. keys of keyset known by the user card)
For computation of response relevant key(s) of EF_{DIK1}, have to be used.

'01'EF_{DIK} number 2. (containing div. keys of new keyset).

Table A.17: Coding of the data field of the COMPUTE LOAD KEY command

Bytes	Description	Length
1	'D8' (Coded according to TS 101 206-3 [2])	1 byte
2	EF_{KEY} type(coded according to TS 101 206-3 [2])	1 byte
3	Key number (coded according to TS 101 206-3 [2])	1 byte
4	L_c of the command sent to the UC (coded according to TS 101 206-3 [2])	1 byte
5	Key file version (only present if key number 0)	1 byte
6	Key length	1 byte
7	Algorithm identifier	1 byte
NOTE: The SM has to check internally that key number, key length, key version and algorithm ID are consistent with the contents of the relevant EF_{KEY} (UC).		

Table A.18: Coding of the COMPUTE LOAD KEY response

Bytes	Description	Length
1 - 16	Enciphered key	16 bytes
17 - 24	MAC	8 bytes

A.5.4.2 COMPUTE MAC

Table A.19: Coding of the COMPUTE MAC command

CLA	As defined in TS 101 206-3 [2] subclause 9.2.
INS	'8A'
P1	'00'
P2	Key number.
L_c field	Length of the data field.
Data field	Data sent to the SM.
L_e field	Maximum length of data expected in response.

Table A.20: Coding of the data field of the COMPUTE MAC command

Bytes	Description	Length
1	INS byte of the following command sent to the UC (Coded according to TS 101 206-3 [2])	1 byte
2	P1 of the following command sent to the UC (coded according to TS 101 206-3 [2])	1 byte
3	P2 of the following command sent to the UC (coded according to TS 101 206-3 [2])	1 byte
4	L_c of the following command sent to the UC (coded according to TS 101 206-3 [2])	1 byte
5 - (4+X)	The data field of the following command sent to the UC (Coded according to TS 101 206-3 [2])	X bytes

Table A.21: Coding of the COMPUTE MAC response

Bytes	Description	Length
1 - X	MAC	X bytes

The data field of the COMPUTE MAC contains the data necessary to compute the MAC used in the following command sent to the UC. A list of all the functions for which a MAC has to be provided by the use of COMPUTE MAC command is given in subclause 8.2.

A.5.4.3 COMPUTE CRYPTOGRAM

Table A.22: Coding of the COMPUTE CRYPTOGRAM command

CLA	As defined in TS 101 206-3 [2] subclause 9.2.
INS	'56'
P1	'00'
P2	Key number.
L_c field	Empty.
Data field	Empty.
L_e field	Maximum length of data expected in response.

Table A.23: Coding of the COMPUTE CRYPTOGRAM response

Bytes	Description	Length
1 - X	Cryptogram	X bytes

A.5.4.4 COMPUTE MAC EW

Table A.24: Coding of the COMPUTE MAC EW command

CLA	As defined in TS 101 206-3 [2] subclause 9.2.
INS	'8C'
P1	'00'
P2	Key number.
L_c field	Length of the data field.
Data field	Data sent to the SM.
L_e field	Maximum length of data expected in response.

Table A.25: Coding of the data field of the COMPUTE MAC EW command

Bytes	Description	Length
1	INS byte of the following command sent to the UC (Coded according to TS 101 206-3 [2])	1 byte
2	P1 of the following command sent to the UC (coded according to TS 101 206-3 [2])	1 byte
3	P2 of the following command sent to the UC (coded according to TS 101 206-3 [2])	1 byte
4	L_c of the following command sent to the UC (coded according to TS 101 206-3 [2])	1 byte
5 - (4+X)	The data field of the following command sent to the UC (Coded according to TS 101 206-3 [2])	X bytes

Table A.26: Coding of the COMPUTE MAC response

Bytes	Description	Length
1 - X	MAC	X bytes

A.5.5 Commands to verify cryptogram or MAC

A.5.5.1 VERIFY MAC

Table A.27: Coding of the VERIFY MAC

CLA	As defined in TS 101 206-3 [2] subclause 9.2.
INS	'8E'
P1	'00'
P2	Key number.
L_c field	Length of the data field.
Data field	Data sent to the SM.
L_e field	Empty.

Table A.28: Coding of the data field of the VERIFY MAC command

Bytes	Description	Length
1	INS byte of the previous command sent to the UC (coded according to TS 101 206-3 [2])	1 byte
2	P1 of the previous command sent to the UC (coded according to TS 101 206-3 [2])	1 byte
3	P2 of the previous command sent to the UC (coded according to TS 101 206-3 [2])	1 byte
4	The L_e of the previous command sent to the UC (coded according to TS 101 206-3 [2])	1 byte
5 - (4+X)	The data field of the previous response sent by the UC (coded according to TS 101 206-3 [2])	X bytes

A list of all the functions for which a MAC has to be verified by the use of VERIFY MAC command is given in subclause 8.3.1.

A.5.5.2 UPDATE (SM)

Table A.29: Coding of the UPDATE (SM) command

CLA	As defined in TS 101 206-3 [2] subclause 9.2.
INS	'5A'
P1	'00'
P2	Key number.
L_c field	Length of the data field.
Data field	Data sent to the SM.
L_e field	Empty.

Table A.30: Coding of the data field of the UPDATE (SM) command

Bytes	Description	Length
1	'B6' (Coded according to TS 101 206-3 [2])	1 byte
2	Record no. (coded according to TS 101 206-3 [2])	1 byte
3	Mode (coded according to TS 101 206-3 [2])	1 byte
4	The L_e of the data field of the previous command (coded according to TS 101 206-3 [2])	1 byte
5 - (4+X)	The data field of the previous READ RECORD STAMPED response sent by the UC (coded according to TS 101 206-3 [2])	X bytes

A.5.5.3 VERIFY CRYPTOGRAM

Table A.31: Coding of the VERIFY CRYPTOGRAM

CLA	As defined in TS 101 206-3 [2] subclause 9.2.
INS	'8E'
P1	'00'
P2	Key number.
L_c field	Length of the data field.
Data field	Data sent to the SM.
L_e field	Empty.

Table A.32: Coding of the data field of the VERIFY CRYPTOGRAM command

Bytes	Description	Length
1 - X	Cryptogram	X bytes

A.5.6 Commands to verify a MAC and compute another

A.5.6.1 INCREASE (SM)

Table A.33: Coding of the INCREASE (SM) command

CLA	As defined in TS 101 206-3 [2] subclause 9.2
INS	'5C'
P1	'00'
P2	Key number.
L_c field	Length of the data field.
Data field	Data sent to the SM.
L_e field	Maximum length of data expected in response.

Table A.34: Coding of the data field of the INCREASE (SM) command

Bytes	Description	Length
1	'34' (coded according to TS 101 206-3 [2])	1 byte
2	Output mode (coded according to TS 101 206-3 [2])	1 byte
3	'00' (coded according to TS 101 206-3 [2])	1 byte
4	The L_e of the data field of the previous response (coded according to TS 101 206-3 [2])	1 byte
5 - (4+X)	The data field of the previous DECREASE STAMPED response sent by the UC (coded according to TS 101 206-3 [2])	X bytes

A.5.6.2 DECREASE (SM)

Table A.35: Coding of the DECREASE (SM) command

CLA	As defined in TS 101 206-3 [2] subclause 9.2.
INS	'5E'
P1	'00'
P2	Key number.
L_c field	Length of the data field.
Data field	Data sent to the SM.
L_e field	Maximum length of data expected in response.

Table A.36: Coding of the data field of the DECREASE (SM) command

Bytes	Description	Length
1	INS byte of the following INCREASE or INCREASE STAMPED command sent to the UC (coded according to TS 101 206-3 [2])	1 byte
2	P1 of the following INCREASE or INCREASE STAMPED command sent to the UC (coded according to TS 101 206-3 [2])	1 byte
3	P2 of the following INCREASE or INCREASE STAMPED command sent to the UC (coded according to TS 101 206-3 [2])	1 byte
4	The L_c of the previous command (coded according to TS 101 206-3 [2])	1 byte
5 - (4+X)	The data field of the following INCREASE or INCREASE STAMPED command sent to the UC (coded according to TS 101 206-3 [2])	X bytes

Table A.37: Coding of the DECREASE (SM) response

Bytes	Description	Length
1 - X	MAC	X bytes

A.6 Status conditions returned by the SM

See subclause 9.3.

A.7 Functions for downloading of keys from SM to UC

For further information see also clause B.9.

Three different cases where keys have to be downloaded exist:

1) Loading of keys in an empty EF_{KEY_MAN} in UC.

   In this case the key used to fulfil the AC is situated in the EF_{KEY_MAN} at the upper level.
   This means that two different sets of diversified keys (contained in two different EF_{DIK}) have to be available in the SM, one containing the key used to fulfil the AC and one containing the keys to be downloaded.

NOTE: To load keys in an empty $EF_{KEY–OP}$ the same scenario has to be used but the relevant key to fulfil the AC is located in the EF_{KEY_MAN} at the same level.

2) Replacing of already existing keys in the EF_{KEY_MAN} of the UC (changing keyset version from n to n+1).

   In this case the key used to fulfil the AC is situated in the EF_{KEY_MAN} containing the keyset version n.
   This means that two different sets of diversified keys (contained in two different EF_{DIK}) have to be available in the SM, one containing the key used to fulfil the AC (in EF_{KEY_MAN} with keyset version n) and one containing the keys to be downloaded (in EF_{KEY_MAN} with keyset version n+1). As long as the relevant key used to fulfil the AC is not replaced, the relevant key of keyset version n is valid.

3) Replacing of already existing keys in the EF_{KEY_OP} of the UC (changing keyset version from n to n+1).

   In this case the key used to fulfil the AC is situated in the EF_{KEY_MAN}.
   This means that two different sets of diversified keys (contained in two different EF_{DIK}) have to be available in the SM, one containing the key used to fulfil the AC (in EF_{KEY_MAN}) and one containing the keys to be downloaded (in EF_{KEY_OP} with keyset version n+1).

NOTE: It is up to the operating system to manage the coexisting of the two different EF_{DIK}, and what kind of keys (OP or MAN) are contained in each of them.

A.7.1 Loading of keys in an empty EF_{KEY_MAN} in UC

a) Selection (by use of the SELECT KEYSET function) of the file at the next upper level (EF_{KEY_MAN} (UC)) containing the master keys for EF_{KEY_MAN} in the UC. These master keys are used to diversify the key used to fulfil the AC for the function LOAD KEY FILE in the UC.

b) Diversification of the master keys and loading of the diversified keys into EF_{DIK1} by the use of DIVERSIFY KEYSET function.

c) Selection (by the use of SELECT KEYSET function) of the file containing the master keys used to diversify the keys which are going to be downloaded.

d) Diversification of the master keys and loading of the diversified keys into EF_{DIK2} by the use of DIVERSIFY KEYSET function.

e) Enciphering of the key (located in EF_{DIK2}) and computation of the MAC. These two operations are performed using the relevant key located in EF_{DIK2} and indicated in the parameters of the function COMPUTE LOAD KEY. This last action has to be repeated for each of the keys to be downloaded.

A.7.2 Changing of already existing keys in the EF_{KEY_MAN} of the UC

a) Selection (by use of the SELECT KEYSET function) of the file (EF_{KEY_MAN} (UC)) containing the master keys (key version n) for EF_{KEY_MAN} in the UC. These master keys are used to diversify the key used to fulfil the AC for the function LOAD KEY FILE in the UC.

b) Diversification of the master keys and loading of the diversified keys into EF_{DIK1} by the use of DIVERSIFY KEYSET function.

c) Selection (by the use of SELECT KEYSET function) of the file (EF_{KEY_MAN} (UC)) containing the master keys (keyset version n+1) used to diversify the keys which are going to be downloaded.

d) Diversification of the master keys and loading of the diversified keys into EF_{DIK2} by the use of DIVERSIFY KEYSET function.

e) Enciphering of the key (located in EF_{DIK2}) and computation of the MAC. These two operations are performed using the relevant key located in EF_{DIK2} and indicated in the parameters of the function COMPUTE LOAD KEY. As long as the relevant key used to fulfil the AC is not replaced, the relevant key of keyset version n is valid.

This last action has to be repeated for each of the keys to be downloaded.

A.7.3 Changing of already existing keys in the EF_{KEY_OP} of the UC

a) Selection (by use of the SELECT KEYSET function) of the file (EF_{KEY_MAN} (UC)) containing the master keys for EF_{KEY_MAN} in the UC. These master keys are used to diversify the key used to fulfil the AC for the function LOAD KEY FILE in the UC.

b) Diversification of the master keys and loading of the diversified keys into EF_{DIK1} by the use of DIVERSIFY KEYSET function.

c) Selection (by the use of SELECT KEYSET function) of the file (EF_{KEY_OP} (UC)) containing the master keys EF_{KEY_OP} (keyset version n+1) used to diversify the keys which are going to be downloaded.

d) Diversification of the master keys and loading of the diversified keys into EF_{DIK2} by the use of DIVERSIFY KEYSET function.

e) Enciphering of the key (located in EF_{DIK2}) and computation of the MAC. These two operations are performed using the relevant key located in EF_{DIK2} and indicated in the parameters of the function COMPUTE LOAD KEY. As long as the relevant key used to fulfil the AC is not replaced, the relevant key of keyset version n is valid.

This last action has to be repeated for each of the keys to be downloaded.

Annex B (informative): Scenarios for SM in the form of an IC card

B.1 Key diversification procedure

The purpose of this operation is for the SM to diversify the individual keyset of the UC. The SM uses its own master key and the diversification data read out from the UC to perform this operation. The diversified keys can then be used by the SM to compute and verify cryptograms to and from the UC.

To perform this functionality the terminal reads out from the user card:

Application identifier AID;

- refer to ISO/IEC 7816-5 [4];
- refer to TS 101 206-3 [2]:

  subclause 6.5: Methods of selecting a file;

  subclause 11.1: standardized applications;

  subclause 10.3: EFDIR;

Relevant EFKEY type;

Key version;

- refer to TS 101 206-3 [2] subclause 9.4.1 SELECT (response in case of EFKEY);

Diversification data.

Minimum set of functions of SM needed to perform this operation:

- SELECT KEYSET;
- DIVERSIFY KEYSET.

Minimum set of functions of UC needed to perform this operation:

- SELECT;
- READ.

Minimum set of data in UC needed to perform this operation:

- key qualifier;
- diversification data.

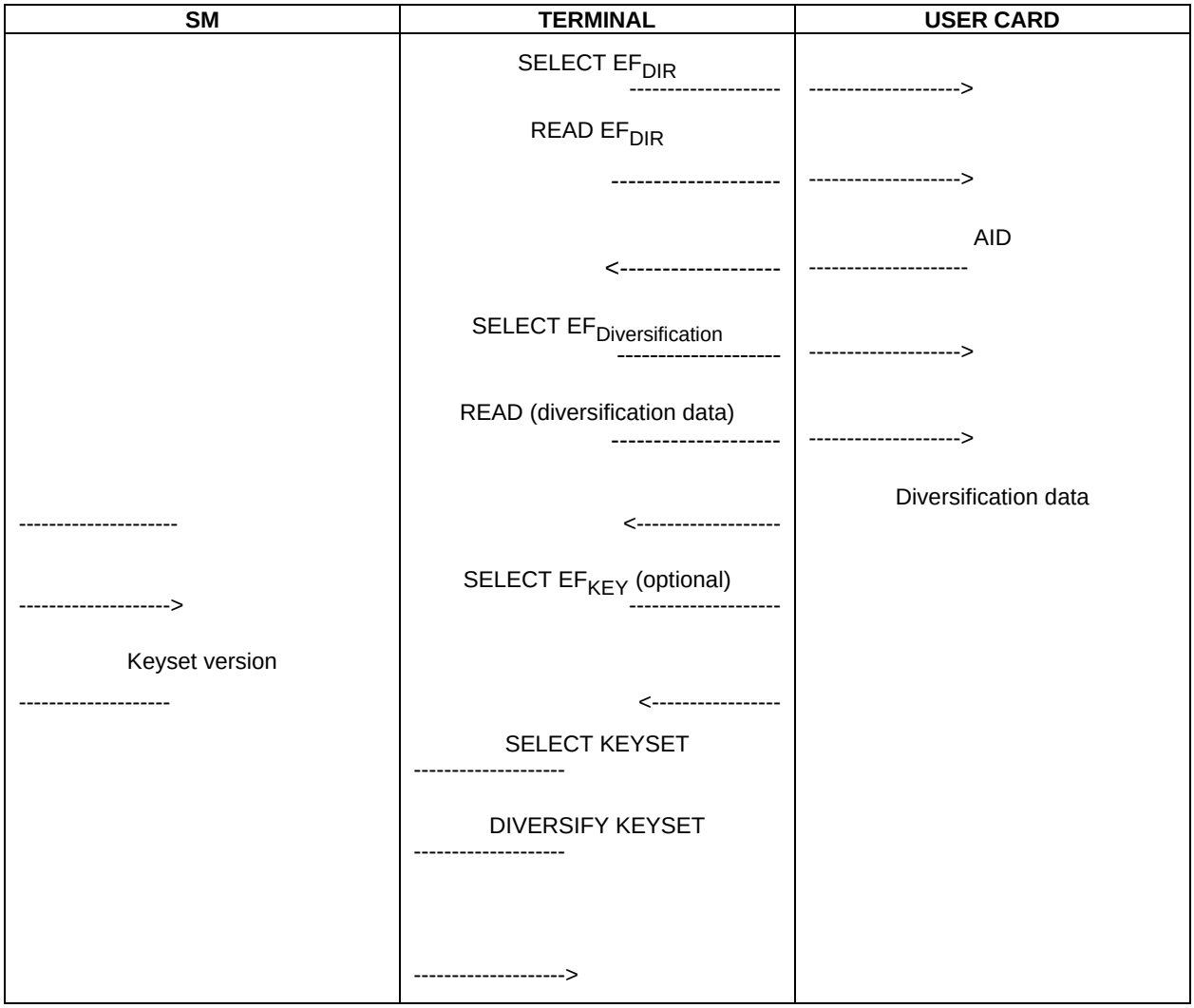

Figure B.1: Scenario for the keyset diversification procedure

B.2 Authentication of UC by the SM

The purpose of this operation is to enable the SM to verify the authenticity of the UC and by doing so to ensure that the application within the UC has been supplied by the entitled authority.

After implementation of the key diversification procedure (see clause B.1), a challenge is provided to the UC.
The answer of the UC is transferred through the terminal to the SM and is checked by the SM.
The result of this check is passed to the terminal by the SM, to be used by it for further decisions.

Minimum set of functions of SM needed to perform this operation:

- VERIFY CRYPTOGRAM (INTERNAL AUTHENTICATION (UC)).

Minimum set of functions of UC needed to perform this operation:

- ASK PARAMETER;
- INTERNAL AUTHENTICATION.

Minimum set of data in UC needed to perform this operation:

- key.

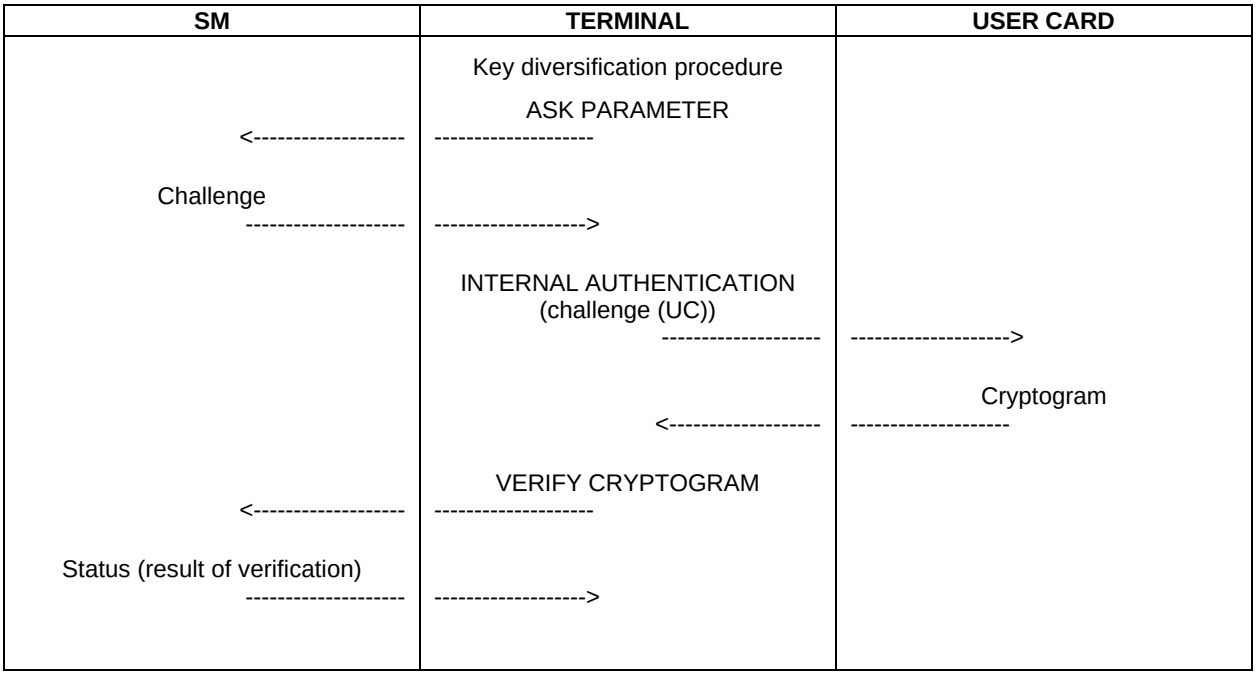

Figure B.2: Scenario for Authentication of UC by the SM

B.3 Authentication of SM by the UC

The purpose of this operation is to enable the UC to verify the authenticity of the SM and the application within the SM, and by doing so to ensure that the application within the SM has been supplied by an authority entitled to use the specific UC.

After implementation of the key diversification procedure (see clause B.1), a challenge created by the UC is given to the SM. The answer of the SM transmitted through the terminal to the UC is checked by the UC. The result is used to prove the AC = AUT of the relevant file of the UC.

Minimum set of functions of SM needed to perform this operation:

- COMPUTE CRYPTOGRAM (EXTERNAL AUTHENTICATION (UC));
- GIVE RANDOM.

Minimum set of functions of UC needed to perform this operation:

- ASK RANDOM;
- EXTERNAL AUTHENTICATION.

Minimum set of data in UC needed to perform this operation:

- key.

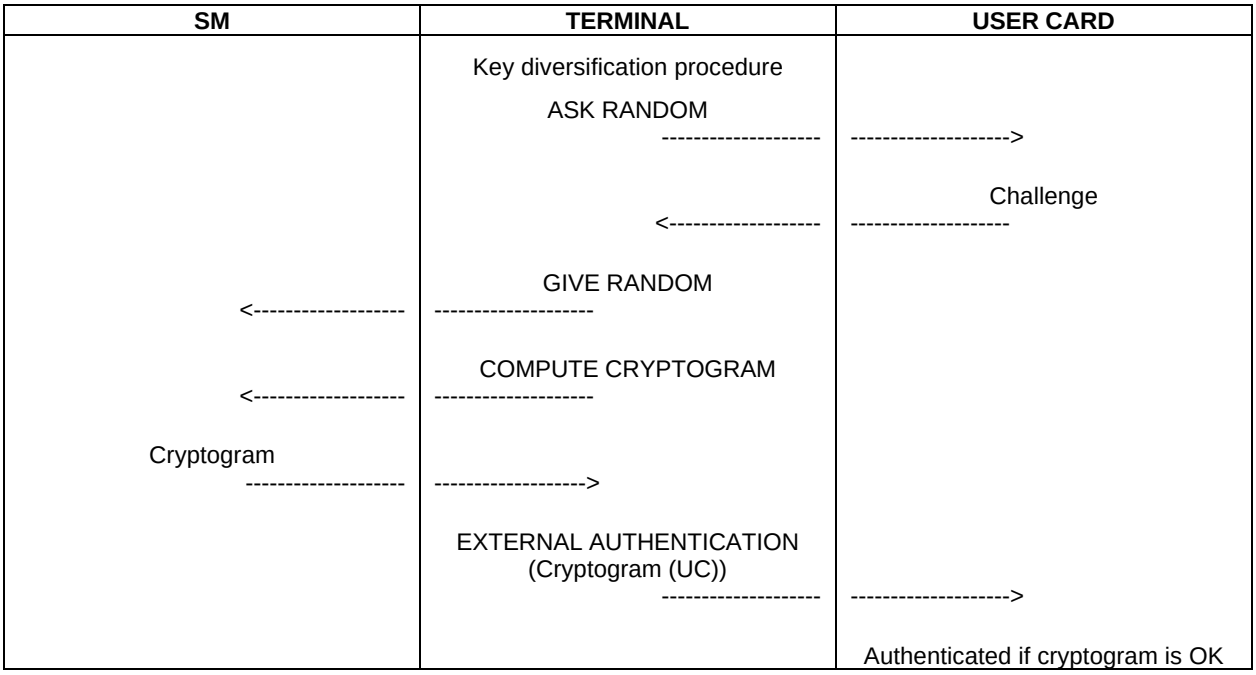

Figure B.3: Scenario for Authentication of SM by the UC

B.4 Compute MAC for protected mode (message authentication)

The purpose of this operation is to generate a cryptogram in the SM that is needed by the terminal to use commands in the protected (PRO) mode of the UC. After implementation of the key diversification procedure (see clause B.1) the terminal sends a command to the SM (including the header of the command to the UC) and if necessary the data that has to be used to compute the cryptogram. The SM checks if it is allowed to use its keys to compute the cryptogram that has been demanded. If the SM is allowed to use its keys, it returns the cryptogram to the terminal which uses it to built the command to the UC.

Minimum set of functions of SM needed to perform this operation:

- GIVE RANDOM;
- COMPUTE MAC.

Minimum set of functions of UC needed to perform this operation:

- ASK RANDOM;
- Command "PRO".

Minimum set of data in UC needed to perform this operation:

- key.

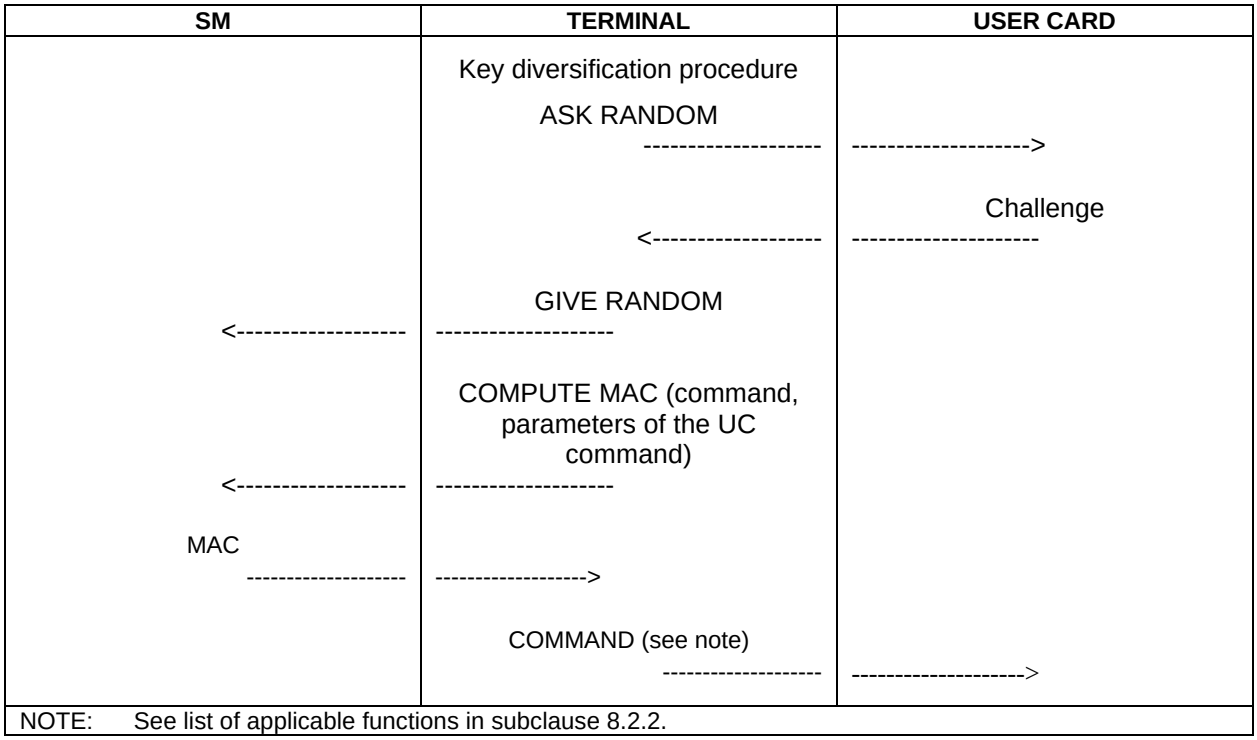

Figure B.4: Scenario for COMPUTE MAC

B.5 Verify MAC command

The purpose of this operation is to verify a cryptogram in the SM. The cryptogram is a result of an internal authentication or the result of a stamped mode command performed in the UC.
This operation is carried out after implementation of the key diversification procedure (see clause B.1).

Minimum set of functions of SM needed to perform this operation:

- ASK PARAMETER;
- VERIFY MAC.

Minimum set of functions of UC needed to perform this operation:

- Command STAMPED;
- GIVE RANDOM.

Minimum set of data in UC needed to perform this operation:

- data read value;
- key.

SM	TERMINAL	USER CARD
	Key diversification procedure	
	ASK PARAMETER	
<----------	----------	
Challenge		
----------	---------->	
	GIVE RANDOM	
	----------	---------->
	COMMAND (See note)	
	----------	---------->
		Response, MAc, status
	<----------	----------
	VERIFY MAC (parameters of the previous UC command/response)	
<----------	----------	
Status (result of verification)		
----------	---------->	
NOTE: See list of applicable functions in subclause 8.3.1.		

Figure B.5: Scenario for VERIFY MAC

B.6 UPDATE (SM)

The purpose of this operation is to update a record in a file of the SM using the read stamped mode of the UC. This operation is carried out after implementation of the key diversification procedure (see clause B.1).

Minimum set of functions of SM needed to perform this operation:

- SELECT;
- ASK PARAMETER;
- UPDATE (SM).

Minimum set of functions of UC needed to perform this operation:

- SELECT;
- READ STAMPED;
- GIVE RANDOM.

Minimum set of data in UC needed to perform this operation:

- data read value;
- key.

SM	TERMINAL	USER CARD
	Key diversification procedure	
<----------	----------	
	ASK PARAMETER	
<----------	----------	
Challenge		
----------	---------->	
	GIVE RANDOM	
	----------	---------->
	SELECT (file)	
	----------	---------->
	READ RECORD STAMPED	
	----------	---------->
		Data, MAC, status
	<----------	----------
	SELECT (file)	
<----------	----------	
	UPDATE (SM) (parameters of the previous READ RECORD STAMPED (UC) command/response)	
<----------	----------	
If verification of the MAC is OK, then update the cyclic file with the data given in UPDATE (SM)		
Status		
----------	---------->	

Figure B.6: Scenario for UPDATE (SM)

B.7 Secure transfer of units from UC to SM

The purpose of this operation is to add an amount to a balance in the SM, only if it was previously decreased from a counter value in a valid UC.
This operation is carried out after implementation of the key diversification procedure (see clause B.1).

Minimum set of functions of SM needed to perform this operation:

- ASK PARAMETER;
- SELECT;
- INCREASE (SM).

Minimum set of functions of UC needed to perform this operation:

- SELECT;
- GIVE RANDOM;
- DECREASE STAMPED.

Minimum set of data in UC needed to perform this operation:

- counter value;
- key.

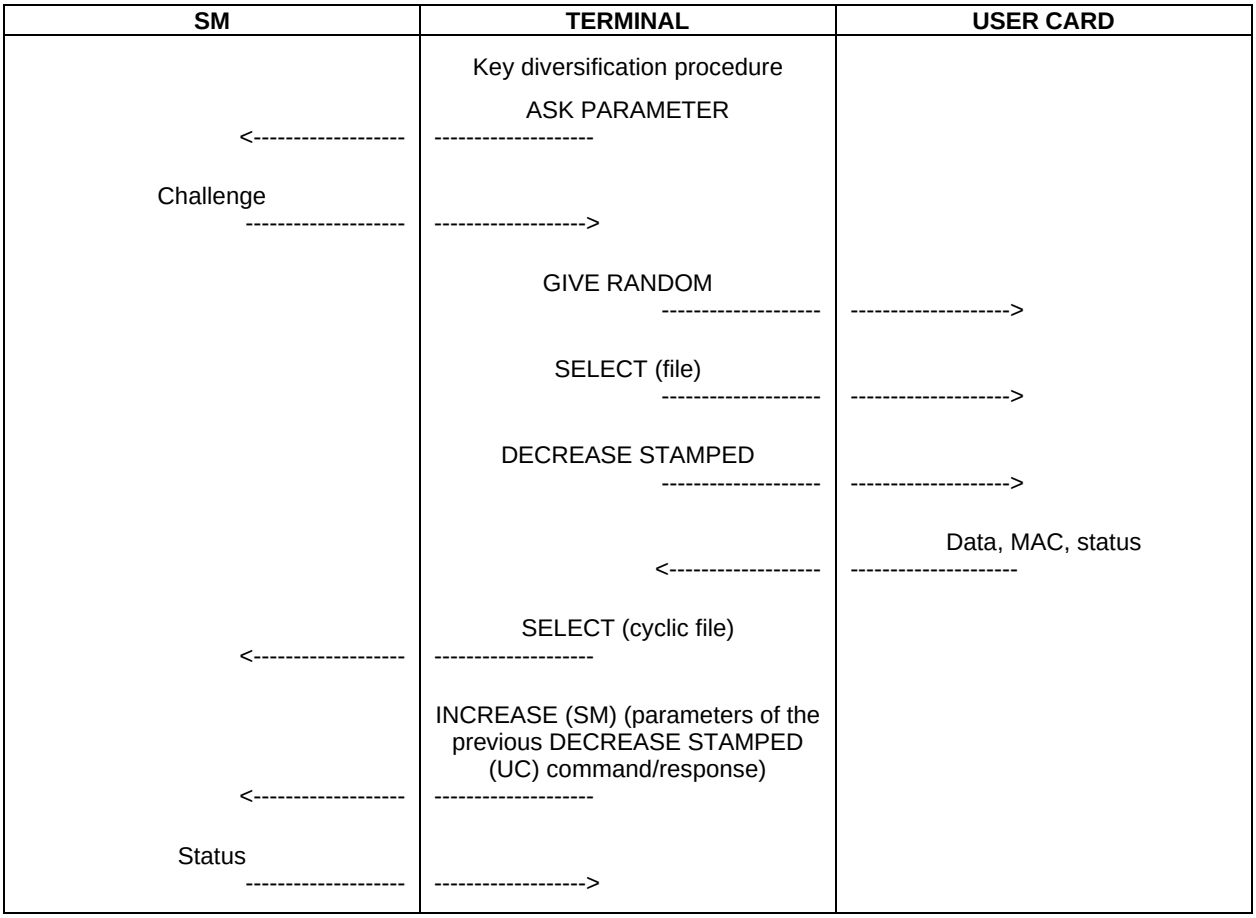

Figure B.7: Scenario for INCREASE (SM)

B.8 Secure transfer of units from SM to UC (reload)

The purpose of this operation is to add an amount to a counter value in a valid UC, only if it was previously decreased from a balance in an authorized SM.
This operation is carried out after implementation of the key diversification procedure (see clause B.1).

Minimum set of functions of SM needed to perform this operation:

- GIVE RANDOM;
- SELECT;
- DECREASE (SM).

Minimum set of functions of UC needed to perform this operation:

- INCREASE;
- ASK RANDOM.

Minimum set of data in UC needed to perform this operation:

- data read value;
- key.

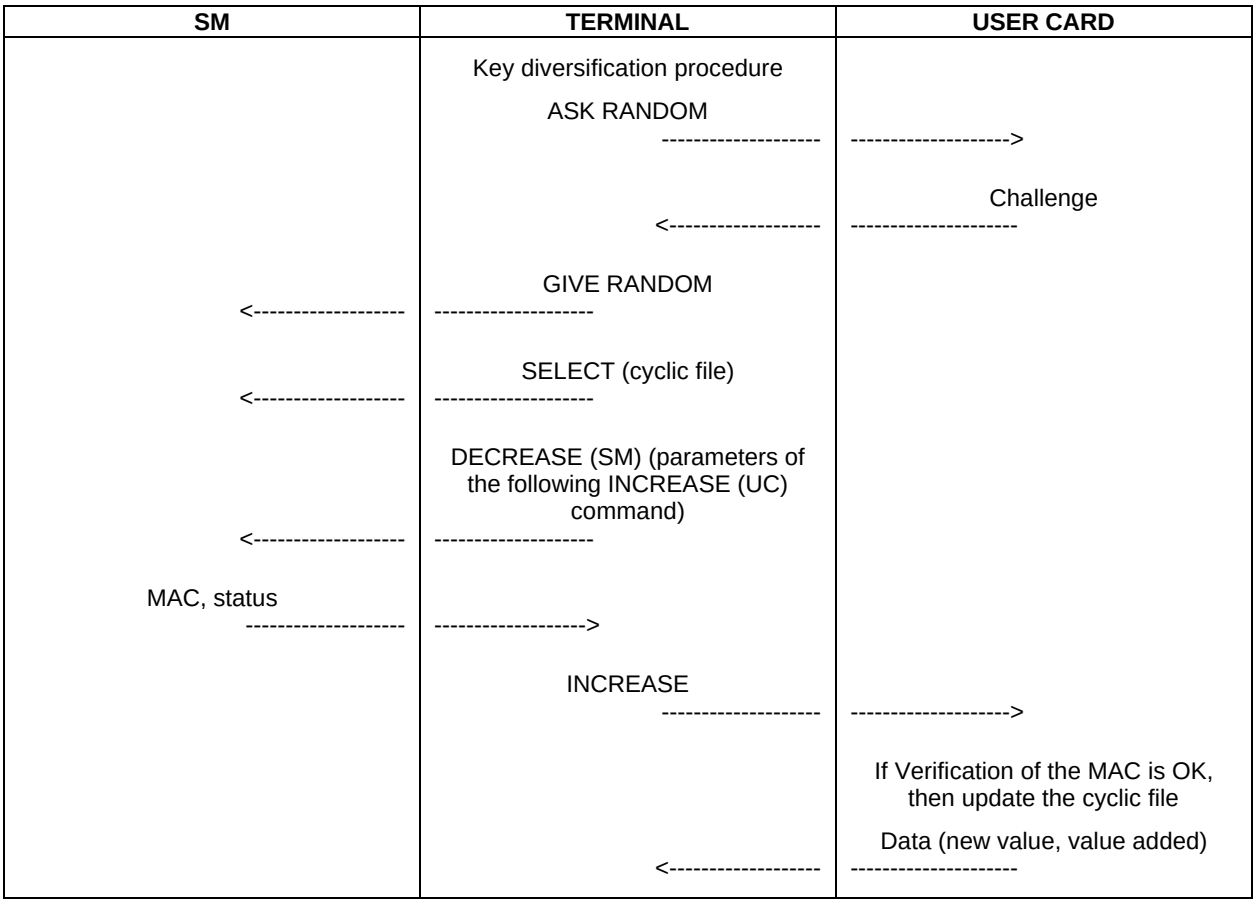

Figure B.8: Scenario for DECREASE (SM)

B.9 Scenarios for downloading keys

B.9.1 Loading of keys in an empty EF_{KEY_MAN} in the UC

The purpose of this operation is to download keys in an empty EF_{KEY_MAN} in UC.

Minimum set of functions of SM needed to perform this operation:

- SELECT KEYSET;
- DIVERSIFY KEYSET;
- GIVE RANDOM;
- COMPUTE LOAD KEY.

Minimum set of functions of UC needed to perform this operation:

- ASK RANDOM;
- LOAD KEY FILE.

Minimum set of data in UC needed to perform this operation:

- AID;
- diversification data;
- key.

SM	TERMINAL	USER CARD
	SELECT KEYSET (master keys of the EF_{KEY_MAN} of the next level above e.g. MF level)	
<----------	----------	
<----------	DIVERSIFY KEYSET (EF_{DIK1}) ----------	
Storage of diversified keys in EF_{DIK1}		
	SELECT KEYSET (master keys of the keys to be loaded e.g. at DF level)	
<----------	----------	
<----------	DIVERSIFY KEYSET (EF_{DIK2}) ----------	
Storage of diversified keys in EF_{DIK2}		
	ASK RANDOM ---------->	---------->
	<----------	Challenge ----------
<----------	GIVE RANDOM ----------	
<----------	COMPUTE LOAD KEY ----------	
Enciphered key + MAC ---------->	---------->	
	LOAD KEY FILE ---------->	---------->
		Storage of the key
	<----------	Status ----------

NOTE: The key linked to the LOAD KEY FILE command shall be the last key to be changed.

Figure B.9: Scenario for loading of one key in an empty EF_{KEY_MAN} in the UC

B.9.2 Replacing of already existing keys in the EF_{KEY_MAN} of the UC

The purpose of this operation is to replace an existing key in EF_{KEY_MAN} of the UC.

Minimum set of functions of SM needed to perform this operation:

- SELECT KEYSET;
- DIVERSIFY KEYSET;
- GIVE RANDOM;
- COMPUTE LOAD KEY.

Minimum set of functions of UC needed to perform this operation:

- ASK RANDOM;
- LOAD KEY FILE.

Minimum set of data in UC needed to perform this operation:

- AID;
- diversification data;
- key.

SM	TERMINAL	USER CARD
	SELECT KEYSET (master keys of the EF_{KEY_MAN} (keyset version n+1))	
<----------	----------	
	DIVERSIFY KEYSET (EF_{DIK1})	
<----------	----------	
Storage of diversified keys in EF_{DIK1}		
	SELECT KEYSET (master keys of the EF_{KEY_MAN} (key set version n))	
<----------	----------	
	DIVERSIFY KEYSET (EF_{DIK2})	
<----------	----------	
Storage of diversified keys in EF_{DIK2}		
	ASK RANDOM	
	----------	---------->
		Challenge
	<----------	----------
	GIVE RANDOM	
<----------	----------	
	COMPUTE LOAD KEY	
<----------	----------	
Enciphered key + MAC		
----------	---------->	
	LOAD KEY FILE	
	----------	---------->
		Storage of the key
		Status
	<----------	----------
NOTE: The key linked to the LOAD KEY FILE command shall be the last key to be changed.		

Figure B.10: Scenario for replacing a key in EF_{KEY_MAN} in the UC

B.9.3 Replacing of already existing keys in the EF_{KEY_OP} of the UC

The purpose of this operation is to replace an existing key in EF_{KEY_OP} of the UC.

Minimum set of functions of SM needed to perform this operation:

- SELECT KEYSET;
- DIVERSIFY KEYSET;
- GIVE RANDOM;
- COMPUTE LOAD KEY.

Minimum set of functions of UC needed to perform this operation:

- ASK RANDOM;
- LOAD KEY FILE.

Minimum set of data in UC needed to perform this operation:

- AID;
- diversification data;
- key.

SM	TERMINAL	USER CARD
	SELECT KEYSET (master keys of the EF_{KEY_OP} (keyset version n+1))	
<----------	----------	
	DIVERSIFY KEYSET (EF_{DIK1})	
<----------	----------	
Storage of diversified keys in EF_{DIK1}		
	SELECT KEYSET (master keys of EF_{KEY_MAN}))	
<----------	----------	
	DIVERSIFY KEYSET (EF_{DIK2})	
<----------	----------	
Storage of diversified keys in EF_{DIK2}		
	ASK RANDOM	
	---------->	---------->
		Challenge
	<----------	----------
	GIVE RANDOM	
<----------	----------	
	COMPUTE LOAD KEY	
<----------	----------	
Enciphered key + cryptogram		
----------	---------->	
	LOAD KEY FILE	
	---------->	---------->
		Storage of the key
		Status
	<----------	----------

Figure B.11: Scenario for replacing a key in EF_{KEY_OP} in the UC

History

Document history		
V1.1.1	August 1997	Identical with the document handed over to CEN. It was not published by ETSI.
V1.2.1	January 1998	Publication

ISBN 2-7437-1873-0
Dépôt légal : Janvier 1998