Advanced Card Systems Ltd.
Card & Reader Technologies

ACOS6 SAM

Reference Manual

info@acs.com.hk
www.acs.com.hk

Table of Contents

Figures

Tables

1.0. Introduction

The purpose of this document is to describe in detail the features and functions of the ACS Smart Card Operating System Version 6 Security Access Module (ACOS6-SAM) developed by Advanced Card System Ltd.

1.1. Features

ACOS6-SAM provides the following features:

- Compliance with ISO 7816 Parts 1, 2, 3, 4
- High baud rate switchable from 9600 to 223,200 bps.
- Full 64 K of EEPROM memory for application data.
- Supports ISO7816 Part 4 file structures: Transparent, Linear fixed, Linear Variable, Cyclic.
- DES / Triple DES capability.
- Hardware based random number generator compliant to FIPS140-2
- Mutual authentication with session key generation.
- Secure Messaging ensures data transfers are confidential and authenticated.
- Secure Access Module pairs with ACOS2, ACOS3, ACOS6 and MIFARE Ultralight C.
- Stores and performs all key operations on ACOS2/3/6 including mutual authentication, encrypted PIN submission, secure messaging, and ePurse commands
- Multilevel secured access hierarchy.
- Anti-tearing done on file headers and PIN commands.

1.2. Technical Specifications

The following are some technical properties of the ACOS6 card:

Electrical

- Operating at 5V DC+/-10% (Class A) and 3V DC +/-10% (Class B)
- Maximum supply current: <10 mA
- ESD protection: ≤ 4 KV

EEPROM

- Capacity: 64 Kbytes (65,536 bytes) including file headers.
- EEPROM endurance: 100K erase/write cycles
- Data retention: 10 years

Environmental

- Operating temperature: -25 °C to 85 °C
- Storage temperature: -40 °C to 100 °C

1.3. Symbols and Abbreviations

3DES	Triple DES
AID	Application / Account Identifier
AMB	Access Mode Byte
AMDO	Access Mode Data Object
APDU	Application Protocol Data Unit
ATC	Account Transaction Counter
ATR	Answer To Reset
ATREF	Account Transaction Reference
CLA	Class byte of APDU commands
COMPL	Bit-wise Complement
COS	Card Operating System
DEC(C, K)	Decryption of data C with key K using DES or 3DES
DES	Data Encryption Standard
DF	Dedicated File
ENC(P, K)	Encryption of data P with key K using DES or 3DES
EF	Elementary File
EF1	PIN File
EF2	KEY File
FCP	File Control Parameters
FDB	File Descriptor Byte
INS	Instruction byte of APDU commands
IV_Seq	Initialization vector with sequence number used in SM-MAC
LCSI	Life Cycle Status Integer
LSb	Least Significant Bit
LSB	Least Significant Byte
MAC	Message Authentication Code
MF	Master File
MSb	Most Significant Bit
MSB	Most Significant Byte
RFU	Reserved for Future Use
RMAC	Retail MAC
SAC	Security Attribute – Compact
SAE	Security Attribute – Expanded
SAM	Security Authentication Module
SCB	Security Condition Byte
SCDO	Security Condition Data Object
SE	Security Environment
Seq#	Sequence number used in SM-ENC

SFI	Short File Identifier
SM-ENC	Secure Messaging with Encryption
SM-MAC	Secure Messaging with MAC
TLV	Tag-Length-Value
$TTREF_C$	Terminal Transaction Reference for Credit
$TTREF_D$	Terminal Transaction Reference for Debit
UL-C	MIFARE Ultralight C
UQB	Usage Qualifier Byte
||	Concatenation

1.4. History of Modification

May 2006	ACOS6-SAM revision 1.00
November 2007	ACOS6-SAM revision 4.00 - Enhancement added for up to 307.2 kbps communication support. - Expanded user storage capacity to 16 Kbyte. - Added bulk encryption with CBC.
November 2008	ACOS6-SAM revision 4.02 - Expanded user storage capacity to 32 Kbyte. - Added FIPS140-2 compliant hardware-based RNG - Added short key external authentication - Global level authentication state kept after selecting different DF. - Encrypt/Decrypt command can perform secure messaging for ACOS3. - Encrypt command can calculate retail-MAC. - ACOS6 Secure Messaging supports SM for Confidentiality (SM-ENC). - Clear card has anti-tearing protection. - Card header special function flag to control additional features. - Activate/Deactivate commands allow changing the card life cycle states. - Remains completely backward compatible to previous ACOS6 versions. - Default ATR changed to TA1=0x95 for increased compatibility.
May 2009	ACOS6-SAM revision 4.06 - Expanded user storage capacity to 64 Kbyte. - Multiple purse file can be used in one DF.
July 2009	ACOS6-SAM revision 4.07 - Supports MIFARE UL-C diversification and authentication.

Table 1: History of Modifications

1.5. Organization

This document is arranged in the following manner:

Section 2.0 gives a general overview of SAM functionality and application usage. It discusses the specialized SAM commands that this card supports.

Section 3.0 discusses the basic card management. The pre-customization of the COS and changing of basic card features including ATR, life cycle, etc, are described in that section.

Section 4.0 is about the card's file system. The card headers and what is contained in the internal security and purse files are discussed.

Section 5.0 is about the security features of the card. The card security options, including file security, mutual authentication, secure messaging, and secure PIN submission is described.

Section 6.0 is the command reference of ACOS6-SAM commands.

Section 7.0 is the command reference of SAM specific commands.

Section 8.0 is a reference of all ACOS6-SAM Status Codes.

Section 9.0 provides the different command flows and interaction with an ACOS2 or ACOS3 client card.

Section 10.0 is a quick start guide and examples of using the ACOS6-SAM as a SAM card.

Section 11.0 gives more example of file system creation of how to personalize and program the ACOS6-SAM. This includes creating different types of files, adding security files, etc.

2.0. ACOS6-SAM

ACOS6-SAM is designed to be used as a general cryptogram computation module or as the security authentication module for ACOS2, ACOS3, ACOS6, MIFARE Ultralight C client cards. The SAM card securely stores the cryptographic keys and use these keys to compute cryptograms for other applications or smart cards. Using this, the keys never leaves the SAM and the system security would be greatly enhanced. The SAM can also perform the authentication procedure and purse MAC computation for the ACOS2/3/6 cards.

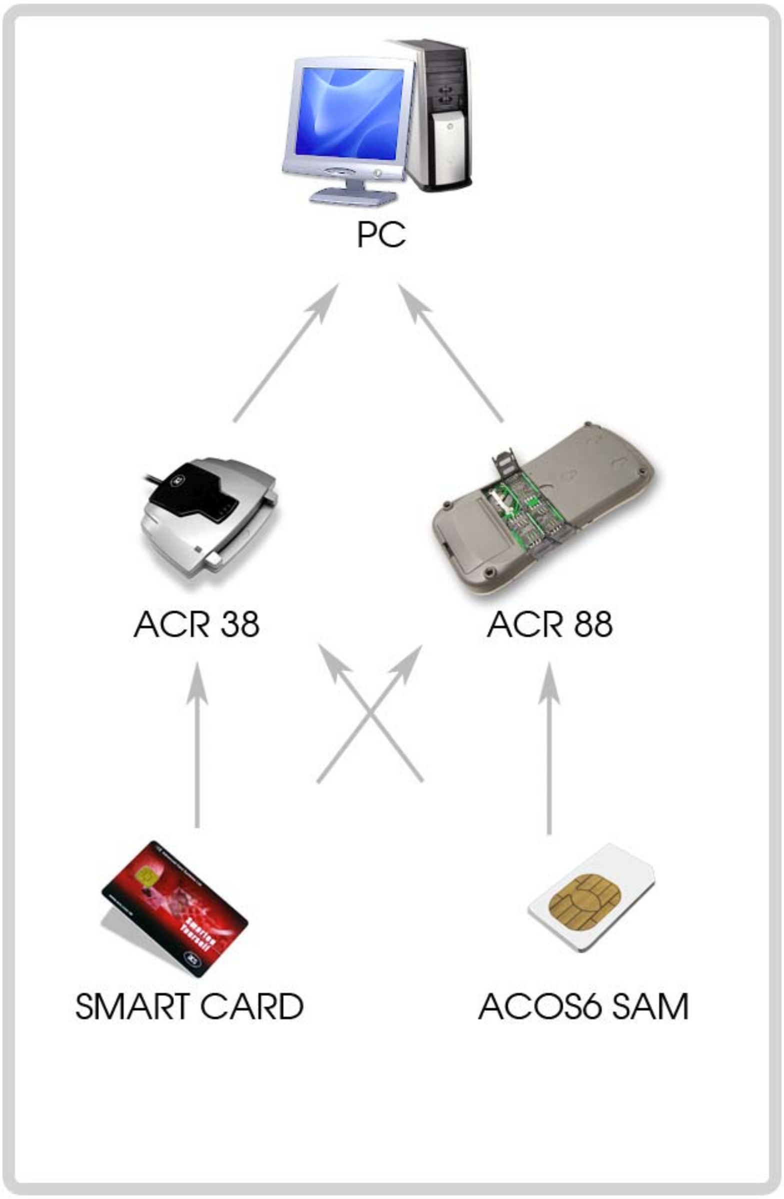


Figure 1: ACOS6-SAM set-up

The ACOS6-SAM can be deployed in any application for these purposes:

- To store and secure the application's DES/3DES master keys.
- To generate and derive application keys based on a set of master keys.
- To perform cryptographic functions with client smart cards.

- To use as a secured encryption module.

When used with ACOS2/3/6 client smart cards, ACOS6-SAM can perform the following functions:

- To initialize the ACOS client card with diversified keys based on the card's unique serial number.
- To perform mutual authentication process, and generate of session key.
- To perform secure messaging with ACOS2/3/6.
- To compute the MAC for the PURSE commands.

When used with MIFARE Ultralight C smart cards, ACOS6-SAM can perform the following functions:

- To initialize the UL-C client card with diversified keys based on the card's unique serial number.
- To perform mutual authentication process.

The programming method of ACOS6-SAM is different from ACOS2/3 cards. It is designed to conform to ISO7816 part 4 file system and command set. To get the application developer up to speed, we have included a quick start guide and sample personalization. The following subsections describe the specific SAM functions.

2.1. Generate Key

This command generates an diversified key, based on a Master Key already presiding in the ACOS6-SAM card. It is generally used in the personalization of client cards in which their diversified application key(s) are based on a master key and the client card's serial number.

2.2. Diversify Key

This command computes the ACOS2/3/6 or UL-C application key (*Target Key*), given the master key index and the client card's serial number. The result will be kept in ACOS6-SAM's memory and will not be divulged to the outside world.

ACOS6-SAM can compute the following diversified target keys:

- ACOS Terminal Key K_T (8 or 16 bytes)	; May represent a Mifare UL-C authentication key.
- ACOS Card Key K_C (8 or 16 bytes)	
- ACOS Session Key K_S (8 or 16 bytes)	
- ACOS Secret Code S_C (8 or 16 bytes)	; may represent a diversified key or the first 8 bytes can represent AC1 AC2 AC3 AC4 AC5 PIN, IC
- ACOS Account Key K_{ACCT} (8 or 16 bytes)	; may represent Debit Key, Credit Key, Certify Key

This command can also load a non-diversified target key to use for bulk encryption.

Note that the Master Keys or non-diversified target key must be a 3DES key since single DES is considered insufficient for most security applications. Depending on the application, the diversified key can be used for single DES operations.

2.3. Encrypt

This command performs and returns Single / Triple DES encryption in ECB, CBC or MAC modes using an diversified key generated by the DIVERSIFY command or a static encryption key in the key file. DES mode and plain text data are supplied by the application.

This command also performs translation of normal ACOS3/6 command to ACOS3/6 secure message command after authentication with the ACOS3 card. The SAM card can perform command data encryption and command MAC in one step for the client ACOS3/6 card SM command. The session key and sequence number will be kept and updated in the SAM card.

2.4. Decrypt

This command performs and returns Single / Triple DES decryption in ECB or CBC modes using a diversified key that was generated by the DIVERSIFY command or a static encryption key in the key file. DES mode and ciphered text data are supplied by the application.

This command can also perform verification of ACOS3/6 secure messaging MAC and decryption of the encrypted response data. It should be call after ENCRYPT command of SM computation mode. The session key and sequence number will be kept and updated in the SAM card.

2.5. Prepare Authentication

This command helps the application perform Mutual Authentication with an ACOS2/3/6 or UL-C client card. The ACOS Card Key (K_C) and Terminal Key (K_T) must have been generated beforehand using the DIVERSIFY command.

The application simply passes the ACOS2/3/6 challenge data to the SAM, and this command will return the required terminal authentication data required by the client ACOS2/3/6 or UL-C which the terminal can send to the client card. The SAM will also generate the Terminal Random data. On success, a session key will be established between client ACOS card and ACOS6-SAM.

2.6. Verify Authentication

This command completes Mutual Authentication with an ACOS2/3/6 or UL-C client card. A successful call to the *Prepare Authentication* command is needed before using this command.

After a successful execution of AUTHENTICATE (and GET REPONSE) command with the client card, the card authentication data from the client card is sent to the AOCS6-SAM card where it will check if the session key generated is correct and complete the mutual authentication process

2.7. Verify ACOS Inquire Account

This command helps the application verify the MAC returned by client card's INQUIRE ACCOUNT. The ACOS purse key (Credit, Debit, Revoke, or Certify keys) must have been generated beforehand using the DIVERSIFY command on the SAM.

This command makes sure that the PURSE information returned by the ACOS client card is authentic.

2.8. Prepare ACOS Account Transaction

This command helps the application perform PURSE commands (CREDIT, DEBIT) with an ACOS2/3/6 client card. The ACOS Purse Key (Credit, Debit, Revoke, or Certify keys) must have been generated beforehand using the DIVERSIFY command on the SAM.

This command will then generate the data (including the MAC) to be sent to the client card.

2.9. Verify ACOS Debit Certificate

This command helps the application verify the DEBIT CERTIFICATE returned by client ACOS3/6 card's DEBIT command. The ACOS Purse DEBIT Key must have been generated beforehand using the DIVERSIFY command on the SAM.

This command ensures that the PURSE information returned by the ACOS3 client card is authentic, and that DEBIT to the card has indeed been executed.

3.0. Card Management

This section outlines the card level features and management functions.

3.1. Anti Tearing

ACOS6-SAM uses an Anti Tearing mechanism in order to protect card from data corruption due to card tearing (i.e., card suddenly pulled out of reader during data update, or reader suffer mechanical failure during card data update). On card reset, ACOS6-SAM looks at the Anti-Tearing fields and does the necessary data recovery. In such case, the COS will return the saved data to its original address in the EEPROM.

3.2. Card Header Block

ACOS6-SAM is a card operating system that has 64K EEPROM. In its initial state (where no file exists), user can access the card header block by using read/write binary with the indicated address.

The Card Header Block contains information about the card. Some card commands' behavior depends on the information in this block. It resides in address EEC0 to EEFF of the EEPROM area using READ/WRITE BINARY. User can access this block only if MF is not present in the card.

It has the following fields and offset:

EEC0 – EEC5: Card ID Number

This is a 6-byte Serial number given by the Card Issuer (or Application Developer). It is used to assign a unique code to each issuer who requests for a unique code. Note that this is different from Card Serial number, which is a unique serial number per card already available in ACOS6-SAM (See Section 6.18 – Get Card Info).

EEC6: ATR Length

If the application wants a customized ATR string, this field will hold the new ATR's length. The customized ATR's length must be > 0 and <= 32.

EEC7: Card Life Cycle Fuse

This field indicates the Life Cycle State of the card. If this byte is not set (0xFF), the card OS will allow user to erase the whole card's contents. At such state, the user is allowed to re-program the card's header block. Please refer to Section 0 for more details on the card's Life Cycle States.

EEC8 - EECA: Random Number Counter – Deprecated

This field was used as a seed in generating random number or challenge data. It is no longer needed due to the availability of a hardware random number generator.

EECB: ACOS2 Record Numbering Mode

For compatibility purposes, this field serves as the RECORD_NUMBERING_FLAG in ACOS2 (in file FF01). If bit5 is 0, record-based files will use index zero in referencing records, otherwise, it will use index one.

EED0 – EEEF: Customized ATR

This field will hold the customized ATR of the card OS. This field is valid if the ATR length (address EEC6) is > 0 and <= 32.

EEF0: Special Function Flags

These flag allows for additional features for ACOS6-SAM revision 4.01 onwards.

b7	b6	b5	b4 b3 b2 b1 b0	Description
0	-	-	-	Key Injection Only Flag
-	0	-	-	Protect Master Key Flag
-	-	0	-	Deactivate Card Enable Flag
-	-	-	1 1 1 1 1	All other values – RFU

Table 2: Special Function Flags

Key Injection Only Flag	After setting this bit to zero and activate the Key file, card must use Get Key and Set Key commands to access Key File. Read and Update records are not allowed after file activation.
Protect Master Key Flag	By setting this bit to zero, Get Key does not allow getting non-diversified keys.
Deactivate Card Enable Flag	Setting this bit to zero allows deactivate command to reset Card Life Cycle Fuse to 0xFF.

3.3. Card Life Cycle States

ACOS6-SAM has the following card states:

1. Pre-Personalization State
2. Personalization State
3. User State

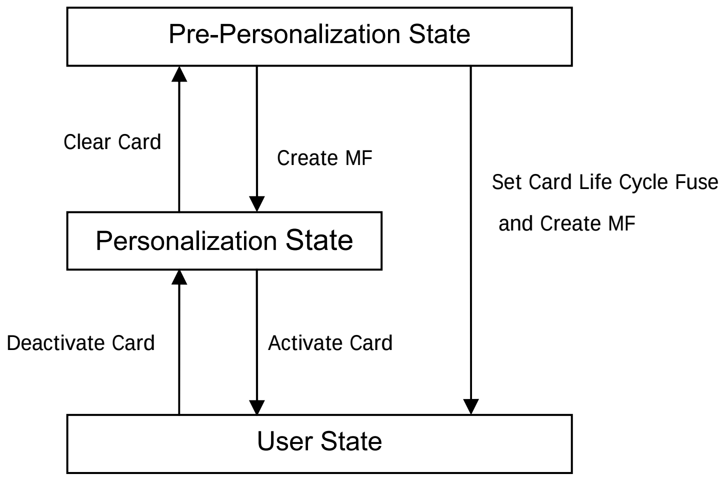


Figure 2: Card life cycle states

Pre-Personalization State – is the initial state of the card. The user is allowed to freely access the card header block (defined in Section 3.2). The card header block can be referenced by its address using the READ BINARY or UPDATE BINARY command.

User can personalize the Card's Header Block as he wishes. Card remains in this state as long as: (1) MF is not created; and (2) the *Card Life Cycle Fuse (address EEC7)* of the *Card Header Block* is 0xFF.

Personalization State – card goes into this state once the MF is successfully created and *Card Life Cycle Fuse* is not blown (still 0xFF). User can no longer directly access the card's memory as in the previous state. User can create and test files created in the card as if in Operational Mode.

User can perform tests under this state and may revert to the Pre-Personalization State by using the Clear Card command.

User State – Card goes into this state once the MF is successfully created and *Card Life Cycle Fuse* is blown. Alternatively, users can use the Activate Card command to go from the personalization state to user state.

The card cannot revert back to previous states when Card Life Cycle Fuse is set (0x00) and bit 5 of Special Function Flags (Deactivate Card Enable Flag) is not set. The Clear Card and Deactivate Card commands are no longer operational.

Typical Development Steps of Card:

1. User personalizes the card's header block using UPDATE BINARY.
2. User then creates his card file structure, starting with MF. DF's and EF's are created and the card's security design is tested at this state. If design flaws are found, user can always return to state 1 using the CLEAR CARD command.
3. Once the card's file and security design is final and tested, perform Clear Card command and blow the *Card Life Cycle Fuse* using the UPDATE BINARY command (write 0x00 to address 0xEEC7).
4. Card goes into Operational Mode, when the MF is created again. User can then re-construct his file system under this state. Card can no longer go back to previous states.

In ACOS6-SAM revision 4.01 and above, user may choose to set the enable Deactivate Card command in card header block. This allows step 3 and 4 to be replaced by the Activate Card command. If the application developer wishes to clear this card, the Deactivate Card command can be used. To control the access to the deactivate card command, an extended security attribute can be set.

3.4. Answer To Reset

After a hardware reset (e.g. power up), the card transmits an Answer-to-Reset (ATR) in compliance with ISO7816 part 3, and it follows the same format as that of ACOS2. ACOS6-SAM supports the protocol type T=0 in direct convention. The protocol function is not implemented.

The following is the default ATR. For full descriptions of ATR options see ISO 7816 part 3.

Parameter	ATR	Description
TS	3B	Direct Convention.
T0	BE	TA1, TB1, TD1 follows with 14 historical characters.
TA1	95	Capable of high-speed communication.
TB1	00	No programming voltage required.
TD1	00	No further interface bytes follow.
14 Historical Characters		

Table 3: Default Configuration of the Answer-to-Reset

The 14 historical characters are composed as the following:

Historical Characters	ATR	Description
T1	41	Indicates ACOS Card
T2	03	Major version
T3	00	Minor version
T4	00	Not used. Compatible with ACOS2
T5	00	
T6	00	
T7	00	
T8	00	
T9	00	
T10	00	
T11	00	
T12	0x	Card Life Cycle State indicator: Bit1: 1=Perso (or pre-perso) State; 0=User State
T13	90	Not used. Compatible with ACOS2
T14	00	

Table 4: Answer-to-Reset Historical Bytes

3.4.1. Customizing the ATR

ACOS6-SAM's ATR can be customized the transmission speed or have specific identification information in the card. The new ATR must be compliant to ISO-7816 Part 3, otherwise the card may become unresponsive and non-recoverable at the next power-up or card reset. Therefore, it is only recommended to change T0 (lower nibble), TA1 and historical bytes.

The transmission speed is determined by the TA1 value in the ATR. More specifically, this field states the different clock rate conversion factor and baud rate adjustment factor. When a smart card reader powers up the card, it will read the ATR at default (low) speed. It will then perform a Protocol and Parameters Selection (PPS) to negotiate a higher transmission rate with the card. Note that the actual baud rate will depend on the card reader's capability and its oscillator. Notice that although ACS has tested the card on all TA1 values with the major readers on the market, there are some readers that may have non-standard timing and may not support the highest speed offered by ACOS6 SAM.

The following is the recommended[1] changes to the ATR:

Parameter	Recommended ATR Value	Description
TS	3B	Direct Convention. *Important: keep this value.*
T0	B*x*	TA1, TB1, TD1 follows with *x* historical characters. Changing the high nibble B is not recommended. See below for the low nibble for the historical byte.
TA1		The following are the reference baud rate[2] and their values:
	96	223,200 bps
	19	192,000 bps
	18	115,200 bps
	95	111,600 bps
	94	55,800 bps
	93	27,900 bps
	92	13,950 bps
	11	9,600 bps
TB1	00	No programming voltage required.
TD1	00	No further interface bytes follow.
Historical bytes: Depends on *x* in T0, which can be 0 to 15 bytes (By setting T0 = B0 to BF respectively). Application developer can use these 15 bytes for personalized identifier information.		

1. Modification of these the non recommended values may make the card permanently unresponsive!
2. Based on an external clock frequency of 3.5712 MHz

Table 5: Customization of the Answer-to-Reset

For Example:

To change the card's ATR to ACOS2, perform the following commands before creating the MF.

```
; Set the desired ATR length to address 0xEEC6, say 13H
< 00 D6 EE C6 01 13
> 9000

; Set the ATR to address 0xEED0
< 00 D6 EE D0 13 3B BE 11 00 00 41 01 38 00 00 00  00 00 00 00 00 00 90 00
> 9000
```

4.0. File System

This section explores the file system of the ACOS6 SAM smart card.

4.1. Hierarchical File System

ACOS6-SAM is fully compliant to ISO 7816 Part 4 file system and structure. The file system is very similar to that of the modern computer operating system. The root of the file is the Master File (of MF). Each Application or group of data files in the card can be contained in a directory called a Dedicated File (DF). Each DF or MF can store data in Elementary Files (EF).

The ACOS6-SAM allows arbitrary depth DF tree structure. That is, the DFs can be nested. Please see Figure 3: below.

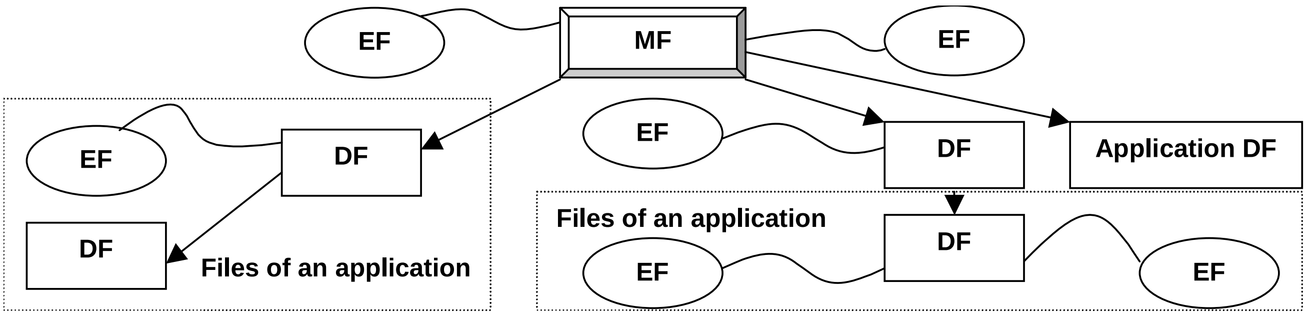


Figure 3: Example of hierarchy of DFs

4.2. File Header Data Structure

ACOS6-SAM organizes the user EEPROM area by files. Every file has a File Header, which is a block of data that describes the file's properties. Knowledge of the file header block will help the application developer accurately plan for the usage of the EEPROM space. The File Header Block consists of the following fields:

File Header	Number of bytes	Applies to
File Descriptor Byte	1	MF / DF / EF
Data Coded Byte	1	MF / DF / EF
File ID	2	MF / DF / EF
File Size	2	EF
Short File Identifier (SFI)	1	MF / DF / EF
Life Cycle Status Integer	1	MF / DF / EF
Security Attribute Compact Length	1	MF / DF / EF
Security Attribute Expanded Length	1	MF / DF / EF
DF Name Length / First Cyclic Record	1	MF / DF / EF
Parent Address	2	MF / DF / EF
Checksum	1	MF / DF / EF
Security Attribute Compact	0-8	MF / DF / EF
Security Attribute Expanded	0-32	MF / DF / EF
Security Environment File ID	2	MF / DF
FCI File ID	2	MF / DF
DF Name	16 max	MF / DF

Table 6: File Header Block

4.2.1. File Descriptor Byte (FDB)

This field indicates the file's type. It can have the following values:

b7	b6	b5	b4	b3	b2	b1	b0	Hex	File type
0	0	1	1	1	1	1	1	3F	**MF**
0	0	1	1	1	0	0	0	38	**DF**
0	0	0	0	0	-	-	-	-	**Working EF**
0	0	0	0	0	0	0	1	01	Transparent (binary) EF
0	0	0	0	0	0	1	0	02	Linear Fixed EF
0	0	0	0	0	1	0	0	04	Linear Variable EF
0	0	0	0	0	1	1	0	06	Cyclic EF
0	0	0	0	1	-	-	-	-	**Internal EF**
0	0	0	0	1	1	0	0	0C	Internal Linear Variable EF (or KEY EF)
0	0	0	0	1	1	1	0	0E	Internal Cyclic EF (Purse EF)

Table 7: File descriptor byte

The size of the File Header block varies depending on the file type. Other values of FDB are considered invalid.

4.2.2. Data Coded Byte (DCB)

ACOS6-SAM does not use this field. It is part of the header to comply with ISO-7816 part 4.

4.2.3. File ID

This is a 16-bit field that uniquely identifies a file in the MF or a DF. Each file under a DF (or MF) must be unique. There are a few pre-defined File ID's. They are:

3F00 - Master File

3FFF - Current DF

FFFF - RFU

A file cannot have an ID of 3FFF and FFFF.

4.2.4. File Size

This is a 16-bit field that specifies the size of the file. It does not include the size of the file header. For record-based EF's, the 1st byte indicates the maximum record length (MRL), while the 2nd indicates the number of records (NOR). For non record-based EF (Transparent EF), the 1st byte represents the high byte of the file size and the 2nd is the low-order byte. For DF's, this field is not used.

4.2.5. Short File Identifier (SFI)

This is a 5-bit value that represents the file's Short ID. ACOS6-SAM allows file referencing through SFI. The last 5 bits of the File ID does not necessarily have to match this SFI. 2 files may have the same SFI under a DF. In such case, ACOS6-SAM will select the one created first.

4.2.6. Life Cycle Status Integer (LCSI)

This byte indicates the life status of the file, as defined in ISO7816 part 4. It can have the following values:

b8	b7	b6	b5	b4	b3	b2	b1	Hex	Meaning
0	0	0	0	0	0	0	1	01	Creation state
0	0	0	0	0	0	1	1	03	Initialization state
0	0	0	0	0	1	-	1	05 or 07	Operational state (activated)
0	0	0	0	0	1	-	0	04 or 06	Operational state (deactivated)
0	0	0	0	1	1	-	-	0C to 0F	Termination state

Table 8: Cycle Status Byte

In Creation / Initialization states, all commands to the file will be allowed by the COS.

In Activated state, commands to the file are allowed only if the file's security conditions are met.

In Deactivated state, most commands to the file are not allowed by the COS.

In Terminated State, all commands to the file will not be allowed by the COS.

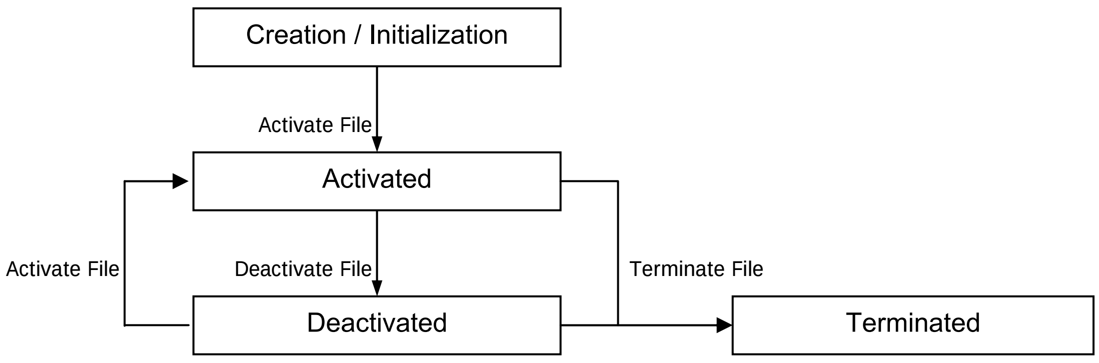


Figure 4: Life cycle status integer

4.2.7. Security Attribute Compact Length (SAC Len)

This byte indicates the length of the SAC structure that is included in the file header below.

4.2.8. Security Attribute Expanded Length (SAE Len)

This byte indicates the length of the SAE structure that is included in the file header below.

4.2.9. DF Name Length / First Cyclic Record

If the file is a DF, this field indicates the length of the DF's Name.

If the file is a Cyclic EF, this field holds the index of the last-altered record.

Otherwise, this field is not used.

4.2.10. Parent Address

2 bytes indicating the physical EEPROM address of the file's parent DF.

4.2.11. Checksum

To maintain data integrity in the file header, a checksum is used by the COS. It is computed by XOR-ing all the preceding bytes in the header. Commands to a file will not be allowed if the file is found to have a wrong checksum.

4.2.12. Security Attribute Compact (SAC)

This is a data structure that represents security conditions for certain file actions. The data is coded in an "AM-SC" template as defined in ISO-7816. The maximum size of this field is 8 bytes. See Section 5.1.1 for more information.

4.2.13. Security Attribute Expanded (SAE)

This is a data structure that represents security conditions for certain card actions. The data is coded differently from SAC, and is also defined in ISO-7816. The maximum size of this field is 32 bytes. See Section 5.1.2 for more information.

For DF files, additional fields are included in the file header:

4.2.14. SE File ID (for DF only)

For DF, this field is made up of 2 bytes containing the File ID of one of its children. That file is known as the Security Environment File, which is processed internally by the COS.

4.2.15. FCI File ID (for DF only)

For DF, this field is made up of 2 bytes containing the File ID of one of its children. That file is known as the File Control Information File, which is processed internally by the COS.

4.2.16. DF Name (for DF only)

For DF, this field is the file's Long Name. Files can be selected through its long name - which can be up to 16 bytes.

4.3. Internal Security Files

The behavior of the COS will depend on the contents of the security-related internal files. When internal files are activated, its READ condition should be set to NEVER. Typically, a DF should have: (1) a Key File to hold PIN codes (referred as EF1) for verification, (2) a Key File to hold KEY codes (referred as EF2) for authentication, and (3) an SE file to hold security conditions.

A Key file is an Internal Linear Variable file. It may contain (1) PIN data structure or (2) KEY data structure.

4.3.1. PIN Data Structure

The PIN is used for VERIFY command. The file that contains PIN records must have an SFI of 1. This file will be referred to as EF1. Each PIN record has the following structure:

	PIN Identifier Byte	Error Counter	PIN
Byte	1	1	16 (max)

Table 9: PIN Data Structure

PIN Identifier Byte: PIN Identifier Byte identifies the PIN number and various options described below:

b7	b6	b5	b4	b3	b2	b1	b0	Description
1	-	-	-	-	-	-	-	The PIN can be altered
-	1	1/0	-	-	-	-	-	The PIN must be encrypted with single DES (b5 = 1) or triple DES (b5 = 0)
-	-	-	x	x	x	x	x	PIN Identifier

Table 10: PIN Identifier Byte

Error Counter: The error counter limits the number of consecutive unsuccessful attempts of PIN submission. This byte is split into two parts. The low nibble indicates the allowed number of retries ($CNT_{Allowed}$) and the high nibble indicates the number of retries left ($CNT_{Remaining}$).

b7	b6	b5	b4	b3	b2	b1	b0
$CNT_{Remaining}$				*$CNT_{Allowed}$*			

Table 11: Error Counter Byte

Each unsuccessful attempt will decrement *$CNT_{Remaining}$*. A successful submission of the PIN number will reset the *$CNT_{Remaining}$* to the *$CNT_{Allowed}$*. If the lower nibble reaches zero, then the PIN is locked and further PIN submission is not possible.

If the error counter is 0xFF, then the error counter is not used and the PIN allows for unlimited number of retries. Hence, the maximum number of retries short of unlimited is 14 or 0x0E.

PIN: The PIN number whose length is between 1 and 16 bytes.

4.3.2. Key data structure

Keys are used for authentication commands. The file that contains the key records must have an SFI of 2. This file will be referred to as EF2. Each KEY record has the following structure:

	Key ID	Key Type	Key Info	Algorithm Reference	Key
Byte	1	1	0, 1, 2 or 3	1	8 or 16

Table 12: Key Data Structure

Key ID: The five LSb's uniquely identifies a key record. If the MSb = 1, the current key record is valid, else it is invalid.

Key Type: This byte indicates the key's capabilities; and also tells the OS how to interpret the Key Info filed.

Bit	Description
b7 - b3	RFU – 0000_b
b3	Key is capable of short key external authentication. Key Info contains *error counter.*
b2	Key is capable of bulk encryption. Key Info is not used.
b1	Key is capable of internal authentication. Key Info contains *usage counter.*
b0	Key is capable of external authentication. Key Info contains *error counter.*

Table 13: Key Type Byte

Internal authentication keys are used for the terminal to authenticate the card. It can also be used by generate key command to generate diversified keys for client cards. Because internal authentication and generate key commands will output encryption results directly given an input. A usage limit can be set to ensure that the key cannot be cracked by cryptanalytic attacks. This is stated in the next section – Key Info.

External authentication keys are used for the card to authenticate the terminal. An error counter will ensure that an authorized terminal cannot keep on accessing the smart card.

A key can also be used to perform *bulk encryption* using ENCRYPT and DECRYPT commands. The *bulk encryption* bit (b2) must be enabled. Bulk encryption allows terminal to directly use the key to do encryption/decryption with plaintext and outputs the result without the key having to be diversified. This feature is useful for the applications that treat the SAM as an secured encryption engine. But it may not be desirable if that key is used for master key.

Key Info: This field depends on the Key Type field. It contains Retry or Usage Counter of the key. Depending on the Key Type field's bits, this field will hold the following: *Internal Authentication Usage Counter* (2 bytes) and External Authentication Error Counter (1 byte).

The *internal authentication usage counter* can limit the number of times a key can be used for internal authentication and generate diversified key. Each execution attempt of the mutual authentication procedure will decrement the usage counter. When the counter reaches zero, the key will become invalid. The counter is two bytes allowing a counter up to 65,534 (0xFFFE). If the counter is 0xFFFF, then the usage of the key is unlimited.

The *external authentication error counter* is the same as PIN Error Counter. Please see Section 4.3.1 for more information. If both external and short key external authenticate are both set, the error counter is shared between the two.

If internal and external and/or short key external authentication bits are set, 2 bytes of usage counter followed by retry counter are in the key info. Either usage counter or retry counter reach zero will render the key invalid.

If key is only used for bulk encryption only, there will be no key info byte.

To summarize the key type and key info relationship, refer to the following table:

Key Type byte:	Key Info
01 , 05, 08, 09, 0C, 0D	1 byte error counter
02, 06,	2 byte usage counter
03, 07,.0A, 0B, 0E, 0F	2 byte usage counter followed by 1 byte error counter
04	No key info byte.
All other values	RFU

Table 14: Relationship between Key Type and Key Info bytes

Algorithm Reference: Algorithm reference states the cryptographic algorithm that is allowed to be used for this key.

Bit	Description
b7 - b3	RFU – 0000000_b
b0	If bit 0 is zero, use triple DES. Else, use single DES.

Table 15: Algorithm Reference byte

Key: The single or triple DES key must have lengths of 8 or 16 bytes respectively.

4.3.3. Security Environment File

A Security Environment (SE) File is an Internal Linear Variable EF that stores SE definitions. Every DF has a designated SE FILE, whose file ID is indicated in the DF's header block. See Section 5.2 for more information.

5.0. Security

File commands are restricted by the COS depending on the target file's (or current DF's) security Access Conditions. These conditions are based on PINs and KEYs being maintained by the system. Card Commands are allowed if certain PIN's or KEY's are submitted or authenticated.

Global PIN's are PINs that reside in a PIN EF (EF1) directly under the MF. Likewise, local Keys are KEYs that reside in a KEY EF (EF2) under the currently selected DF. There can be a maximum of: 31 Global PINs, 31 Local PINs, 31 Global Keys, and 31 Local Keys at a given time.

5.1. File Security Attributes

Each file (MF, DF, or EF) has a set of security attributes set in its headers. There are two types of security attributes Security Attribute Compact (SAC) and Security Attribute Expanded (SAE).

5.1.1. Compact (SAC)

The SAC is a data structure that resides in each file. It indicates what file actions are allowed on the file, and what conditions need to be satisfied for each action. It starts with the AM byte, followed by SC byte(s). The AM byte is bit- coded as follows:

	b7	b6	b5	b4	b3	b2	b1	b0
For MF/DF	Not used	Delete Self	Terminate	Activate	Deactivate	Create DF	Create EF	Delete child
For EF	Not used	Delete Self	Terminate	Activate	Deactivate	-	Update	Read
For Key EF	Not used	Delete Self	Terminate	Activate	Deactivate	-	Set Key	Get Key

Table 16: Access Mode Byte

The number of "1" bits in the AM byte determines the number of SC byte(s) that follow. Bits that are "0" imply that the associated action to the file has no condition (free). Bits with "1" means that a corresponding SC byte exists in the SAC, and that the associated action is allowed only if the SC condition is satisfied.

The SC byte is interpreted as follows:

b7	b6	b5	b4	b3	b2	b1	b0	Meaning
0	0	0	0	0	0	0	0	No condition (always)
1	1	1	1	1	1	1	1	Never
-	-	-	-	0	0	0	0	RFU
-	-	-	-	Not all equal				Security environment identifier (SEID byte) from one to fourteen
-	-	-	-	1	1	1	1	RFU
0	-	-	-	-	-	-	-	At least one condition
1	-	-	-	-	-	-	-	All conditions
-	1	-	0	-	-	-	-	Secure messaging (MAC only – according to Section 5.5)
-	1	-	1	-	-	-	-	ISO Secure messaging (Encryption and MAC – according to Section 5.6)

Table 17: Security Condition Byte

The SE record is found in the SE file - whose ID is specified in the current DF's header. If the SE file is not found, or has incompatible file attributes (internal LV, MRL, NOR, etc.), then the command is denied.

Example: a DF with SAC of 7D 02 03 04 FF FF 02$_H$ means:

AM: $7D_H$ -> has 6 "1" bits (01111101), 6 SC bytes follow; all file actions except "create child EF" is present, "create child EF" is therefore free.

SC: 02 03 04 FF FF 02_H

- allow Delete Self if SE#2 is satisfied
- allow Terminate if SE#3 is satisfied
- allow Activate if SE#4 is satisfied
- do not allow Deactivate
- do not allow Create child DF
- allow Delete Child DF if SE#2 is satisfied

5.1.2. Expanded (SAE)

The SAE is a data structure that resides in each file. It tells the COS whether or not to allow file commands to proceed. SAE is more general compared to SAC. The format of SAE is an access mode data object (AMDO) followed by one or more security condition data objects (SCDO).

<AMDO$_1$><SCDO$_1$> <AMDO$_2$><SCDO$_2$> … <AMDO$_n$><SCDO$_n$>

The COS will check if the command (APDU) falls under the SAE's <AMDO$_i$> if it does it will check the corresponding <SCDO$_i$>.

AMDO: The value field of this data object specifies the command description: CLA INS P1 P2. The low nibble of the TAG indicates which command byte(s) follow:

b7	b6	b5	b4	b3	b2	b1	b0	Meaning
1	0	0	0	x	x	x	x	The command description includes
1	0	0	0	1	-	-	-	CLA
1	0	0	0	-	1	-	-	INS
1	0	0	0	-	-	1	-	P1
1	0	0	0	-	-	-	1	P2

Table 18: Access Mode Data Object (AMDO)

SCDO: May have the following Tags (and Length):

Tag	Length	Value	Meaning
90_H	00_H	-	Allow
97_H	00_H	-	Never
$9E_H$	01_H	SC Byte	SC Byte the same as SAC
$A4_H$	Var.	Authentication Template	Allow if AT condition is satisfied
$A0_H$	Var.	SC DO	One or more SC DO follows where at least 1 condition is met
AF_H	Var.	SC DO	One or more SC DO follows where all conditions are met

Table 19: Security Condition Data Object (SCDO)

Example 1: An SAE of: 86 02 22 F2 97 00_H means:

<AMDO>: 86 02 22 $F2_H$ -> command with INS=22_H and P1 =$F2_H$

<SCDO>: 97 00_H -> never

Hence, the SAE tells the COS not to allow APDU commands with INS=22_H and P1=$F2_H$

Example 2: An SAE of: 84 01 22 A4 06 83 01 81 95 01 08_H means:

<AMDO>: 84 01 22_H -> command with INS=22_H

<SCDO>: A4 06 83 01 81 95 01 08_H -> allow command if local PIN #81 is submitted

Hence, the SAE tells the COS to allow commands with INS=22_H only if the local PIN#1 is verified.

* In ACOS6-SAM, if one <AMDO> matches the command APDU, the COS will not check the subsequent <AMDO>'s.

5.2. Security Environment

Security conditions are coded in an SE File. Every DF has a designated SE FILE, whose file ID is indicated in the DF's header block. Each SE record has the following format:

<SE ID Template> <SE Authentication Template>

SE ID Template: The SE ID Template is a mandatory data object whose value states the identifier that is referenced by the SC byte of the SAC and SAE. The Tag is 0x80 with the length of 0x01.

SE Authentication Template: The Authentication Template (AT) defines the security condition that must be meant for this SE to be satisfied. The security conditions are either PIN or Key authentications.

The tag of AT is 0xA4 and its value contains one more data objects.

Tags recognized within AT:

Tag	Length	Meaning
83_H	01_H	It indicates the identifier of which Key or PIN to use. If its MSB is 1, use EF1 / EF2 under the currently selected DF. Otherwise, use the EF1 / EF2 under the MF.
95_H	01_H	Indicates what action to perform: Authenticate or Verify. The value of this tag has the following meaning: bit7 = AUTHENTICATE the referenced key in tag 83 bit3 = VERIFY the referenced PIN in tag 83

Table 20: Authentication Template Data Objects

Example #1: If the following record is specified in the SE File:

80 01 03 A4 09 83 01 84 83 01 81 95 01 80_H

It has the following meaning:

- The SE ID is 03_H (SE#3).
- AT is: A4 09 83 01 84 83 01 81 95 01 80_H; This contains two 83_H tags inside. This means:
  - allow command if local KEY 84_H is authenticated;
  - allow command if local KEY 81_H is authenticated;
- SE conditions are referenced by an SC byte (in the SAC field of the target file). If the SC byte's MSb is set, then allow command if both conditions are satisfied. If the SC byte's MSB is not set, allow command if at least one condition is satisfied.

Example #2: If the following record is specified in the SE File:

80 01 05 A4 09 83 01 84 83 01 01 95 01 88_H

It has the following meaning:

- The SE ID is 05_H (SE#5);.
- AT is: A4 09 83 01 84 83 01 81 95 01 88_H; contains 2 conditions inside it.
- 1st condition in AT is: 83 01 84 95 01 88_H -> allow command if local KEY 84_H is authenticated AND if local PIN 84_H is verified;
- 2nd condition in AT is: 83 01 01 95 01 88_H -> allow command if global KEY 01_H is authenticated AND if global PIN 01_H is verified;
- SE conditions are referenced by an SC byte (in the SAC field of the target file). If the SC byte's MSB is set, then allow command if both conditions are satisfied. If the SC byte's MSB is not set, allow command if at least one condition is satisfied.

5.3. Mutual Authentication

Mutual Authentication is a process in which both the card and the card-accepting device verify that the respective entity is genuine. A *Session Key* is the result of a successful execution of mutual authentication. The session key is only valid during a *session*. A session is defined as the time after a successful execution of the mutual authentication procedure and a reset of the card or the execution of another mutual authentication procedure.

ACOS6-SAM uses the same authentication mechanism as ACOS2. Hence it is backward compatible. As in ACOS2, the mutual authentication procedure involves two main commands, GET CHALLENGE and MUTUAL AUTHENTICATE. GET RESPONSE is also necessary after the MUTUAL AUTHETNICATE command.

5.3.1. Mutual Authentication Procedure

To provide maximum flexibility, ACOS6-SAM can use two different pairs of DES keys: A *terminal key* K_T and a *card key* K_C. These pair of keys should both be onboard, and they both should either be 8 bytes (for single DES) or both 16 bytes (for triple DES)

A random number generator is onboard ACOS6-SAM to generate a *card random number, RND_C*, in response of the GET CHALLENGE command.

Please follow the figure below for the detailed steps of mutual authenticate procedure.

Terminal		ACOS6-SAM
GET CHALLENGE 00 84 00 00 08	➔ ➔ (←)	Generate and return card challenge RND_C
1. Generate terminal challenge RND_T 2. Compute R_1 = ENC (RND_C, K_T) 3. Compute Session Key K_S 4. Send terminal response and card challenge EXTERNAL AUTHENTICATE (R_1, RND_T) 00 82 K_C K_T 10 R_1 RND_T	➔ ←	1. Check if K_T index is valid and is capable of EXT AUTH. 2. Check if K_C index is valid and is capable of INT AUTH. 3. Verify if R_1 = ENC (RND_C, K_T) 4. Compute K_S 5. Compute card response. R_2 = ENC (RND_T, K_S); Return 6108
GET RESPONSE 00 C0 00 00 08 Verify card authentication Check if R_2 = ENC (RND_T, K_S)	➔ ←	Return card response R_2

Figure 5: Mutual authentication procedure

In the procedure, the encryption, ENC, is either DES or 3DES depending on the Key used.

5.3.2. Session Key Computation

The Session Key is formed through Mutual Authentication between card and terminal. The Session Key's formula depends on the Algorithm Reference field of the KEY. If both Internal key (card key) and external key (terminal key) have Algorithm Reference field of 0, then a 16-byte session key will be formed (Triple DES), otherwise, the session key formed is 8 bytes long (Single DES).

For 16-byte session key K_S, triple DES is used and K_S is generated as follows:

$K_S \quad = \quad K_{sL} \,||\, K_{sR}$

Where K_{sL} is the first 8-byte (or the left half) of the session key:

$K_{sL} \quad = \quad \text{3DES (3DES (}RND_C, K_C\text{)}, K_T\text{)}$

And K_{sR} is the second 8-byte (or right half) of the session key:

$K_{sR} \quad = \quad \text{3DES (}RND_T, \text{REV (}K_T\text{))}$

The variables are the same as that in Section 5.3.1. It is repeated here for completeness:

RND_C:	8-byte Card challenge
RND_T:	8-byte Terminal challenge
K_C:	16-byte Card Key (if key length < 16, 0xFF will be padded)
K_T:	16-byte Terminal Key (if key length < 16, 0xFF will be padded)

The operation REV (K_T) is the exchange of the left and right half of K_T. That is:

$\text{B7 B6 B5 B4 B3 B2 B1 B0} \rightarrow \text{B3 B2 B1 B0 B7 B6 B5 B4}$

For 8-byte session key K_S, single DES is used and K_S is generated as follows:

$K_S \quad = \quad \text{DES (DES (}RND_C, K_C\text{) XOR } RND_T, K_T\text{)}$

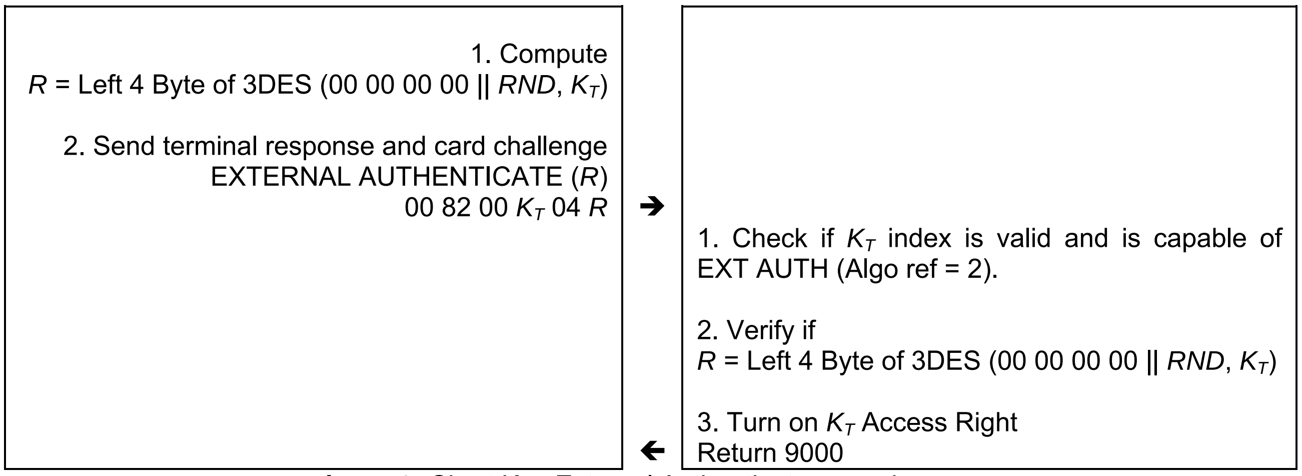
The variables are the same as above except:

K_C:	8-byte Card Key (if key length > 8, only the 1st 8 bytes will be used)
K_T:	8-byte Terminal Key (if key length > 8, only the 1st 8 bytes will be used)

5.4. Short Key External Authentication

Short key external authentication uses a card challenge and terminal response method to gain authorization to the card. This allows for shorter external authentication or one-time-password that is more optimal for human input.

Terminal		ACOS6
GET CHALLENGE 00 84 00 00 04	$\rightarrow$ $\leftarrow$	Generate and return card challenge *RND*


1. Compute
R = Left 4 Byte of 3DES (00 00 00 00 || RND, K_T)

2. Send terminal response and card challenge
EXTERNAL AUTHENTICATE (R)
00 82 00 K_T 04 R →

1. Check if K_T index is valid and is capable of EXT AUTH (Algo ref = 2).

2. Verify if
R = Left 4 Byte of 3DES (00 00 00 00 || RND, K_T)

3. Turn on K_T Access Right
← Return 9000

Figure 6: Short Key External Authenticate procedure

5.5. Secure Messaging for Authenticity (SM-MAC)

ACOS6-SAM supports two types of Secure Messaging - *Secure Messaging for Authenticity (SM-MAC)* and *Secure Messaging for Confidentiality (SM-ENC)*. This section discusses SM for Authentication while Section 5.6 discusses SM for confidentiality.

SM for Authentication allows data and command that is transferred into the card and vise versa to be authenticated. This ensured the command is not modified or replayed. Data blocks sent from the sender to the recipient are appended with 4 bytes of MAC. The receiver then verifies the MAC before proceeding with the operation. Before performing SM, both parties must first have a session key by performing mutual authentication in Section 5.3.

Secure messaging applies to various ISO-in and ISO-out commands. The following is a list of those commands:

ISO-in	**ISO-out**
Create File	
Update Binary	
Update Record	
Append Record	Read Binary
Activate File	Read Record
Deactivate File	
Terminate DF	
Terminate EF	
Delete File	

Table 21: Secure Messaging Commands

5.5.1. SM for ISO-in Command

In an original ISO-in command, the command data looks like this:

	CLA	INS	P1	P2	P3 = n	Command Data
Bytes	1	1	1	1	1	n

With SM, the following is the format of the ISO-in SM command:

	CLA*	INS	P1	P2	P3*	Command Data	$RMAC_{cmd}$
Bytes	1	1	1	1	1	n	4

The CLA* in SM-MAC command is CLA OR 0x04. The P3* field should be n plus 4 to accommodate the 4-byte MAC. The $RMAC_{cmd}$ field is the result of the computation of the Retail MAC operation for Secure Messaging (RMAC). SM-MAC is described in Section 5.5.3. The RMAC takes the following as input:

$RMAC_{cmd}$ = RMAC (CLA* INS P1 P2 P3* CommandData Padding, *IV_Seq*)

The *padding* is needed in RMAC to make the input to the DES algorithm align to a multiple of 8 bytes. The padding is a 0x80 byte followed by 0 to 7 0x00 to make the length of the data to be authenticated equal to a multiple of 8. If the data to be authenticated is in a multiple of 8, then a 0x80 byte is appended followed by 7 0x00 for the last block.

The *IV_Seq* is the initialization vector of the RMAC computation. It is the concatenation of first four bytes of the 8-byte card random number RND_C obtained by a GET CHALLENGE command used during mutual authentication; followed by 00 00_H and a 2 byte SM-MAC sequence number. The SM-MAC sequence number starts at 00 00_H from the last GET CHALLENGE and incremented everytime a SM-MAC command is executed. Calling GET CHALLENGE again would reset the *IV_Seq*.

In ACOS6-SAM, the maximum P3* for SM commands is 132. Therefore, the actual length of data being transmitted between terminal and card is 128 or less.

5.5.2. SM for ISO-out Command

For an ISO-out command, the command has this format:

	CLA*	INS	P1	P2	P3*
Bytes	1	1	1	1	1

The CLA* in the ISO-out command is CLA OR 0x04. The P3* field should be *n* (the size of the original data expected) plus 4 to accommodate the 4-byte $RMAC_{rsp}$. Therefore, P3* should be at least 4 bytes.

The following is the output of a successful execution of the command,

	Response Data	$RMAC_{rsp}$	90 00
Bytes	*n*	4	2

The $RMAC_{rsp}$ field is the result of the computation of the SM-MAC and it takes the following as input.

$RMAC_{rsp}$ = RMAC (CLA* INS P1 P2 P3* ResponseData Padding, *IV_Seq*)

Padding and *IV_Seq* is the same as Section SM for ISO-in Command.

5.5.3. Computing the Secure Messaging Retail MAC (RMAC)

The RMAC for SM-MAC is computed using ISO/IEC 9797-1 CBC-MAC algorithm 3 (ANSI Retail MAC) with DES block cipher according to the figure below.

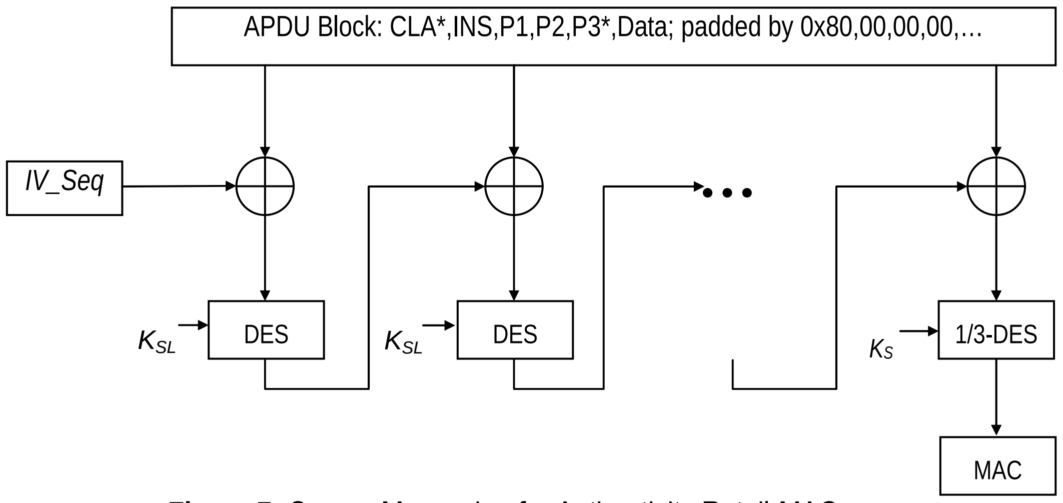

APDU Block: CLA*,INS,P1,P2,P3*,Data; padded by 0x80,00,00,00,…
IV_Seq
K_{SL}
DES
K_{SL}
DES
K_S
1/3-DES
MAC

Figure 7: Secure Messaging for Authenticity Retail-MAC

1. The Initial Vector *IV_Seq* is first 4 bytes of the 8-byte Challenge Data generated by the card. Issue a GET CHALLENGE command prior to an SM command execution. The last 4 bytes of *IV_Seq* will be 00 00_H concatenated with a two byte sequence number.
2. In each round of the DES encryption, K_{SL} is the left half of the session key K_S if K_S 16 bytes long. If K_S is 8-byte long, then K_{SL} refers to K_S.
3. The APDU transmission block is MAC'ed in a group of 8-byte inputs. The last block is padded with 0x80 followed by zeroes. If the APDU block is a multiple of 8, the last block will then be: 0x80, 00, 00, 00, 00, 00, 00, 00.
4. After computation, the 4 MSB of SM-MAC is appended to the APDU block. That is used for transmission for the ISO-in or ISO-out command in Sections 5.5.1 and 5.5.2 respectively.
5. After every computation of SM-MAC, the sequence number in IV_Seq is incremented.

5.6. Secure Message for Confidentiality (SM-ENC)

ACOS6 Version 4.02 and above supports ISO secure messaging (SM). Secure messaging ensures data transmitted between the card and terminal/server is secured and not susceptible to eavesdropping, replay attack and unauthorized modifications. Conditions of requiring secure messaging can be set at the appropriate security conditions byte in Section 5.1.1. Almost all the command can also use secure messaging initiated by the terminal. The commands that do not accept secure messaging are GET CHALLENGE, MUTUAL AUTHENTICATION and GET RESONSE.

The SM employed in this section both encrypts and signs the data transmitting into and out of the card. The card will interprets the terminal command is in SM mode if the CLA of the command has the secure messaging bits set.

Sections 5.6.1 to 5.6.7 discuss the constructs used in secure messaging. Section 5.6.8 details the exact constructs of secure messaging commands and response. Section 5.6.9 lists the secure messaging return codes.

5.6.1. Notations

To describe the two SM modes adequately, there are some notations to be introduced in this section. The following are the possible ISO7816 part 3 communications protocol command and response pairs.

Commands:

a. Without command data:

	CLA	INS	P1	P2	P3
Bytes	1	1	1	1	1

b. With command data

	CLA	INS	P1	P2	P3 = n	Data in
Bytes	1	1	1	1	1	n

Responses:

a. Without response data

	SW1-SW2
Bytes	2

b. With response data

	Response	SW1-SW2
Bytes	n	2

- The class (CLA) byte stated above does not have the SM bits set.
- The modified CLA* for SM in Section 5.6.8 has the SM bits b3 and b2 set to 1. That is, the original CLA XOR $0C_H$.
- P3 is the normal length of the command data.
- P3* will be the length of the command data under SM.

The following figure shows the command transformation of a command without data (ISO-OUT) to a secure messaging APDU command:

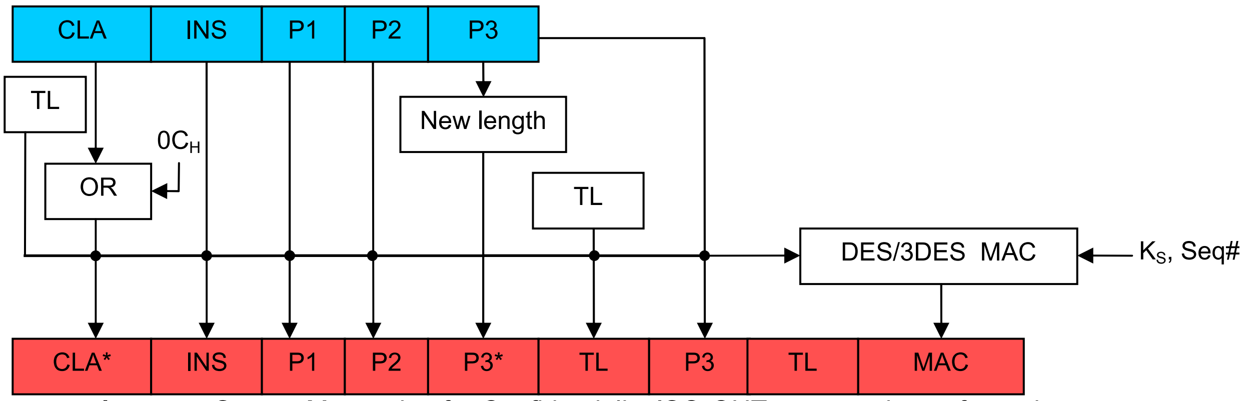


Figure 8: Secure Messaging for Confidentiality ISO-OUT command transformation

The following figure shows the command transformation of a command with data (ISO-IN) to a secure messaging APDU command:

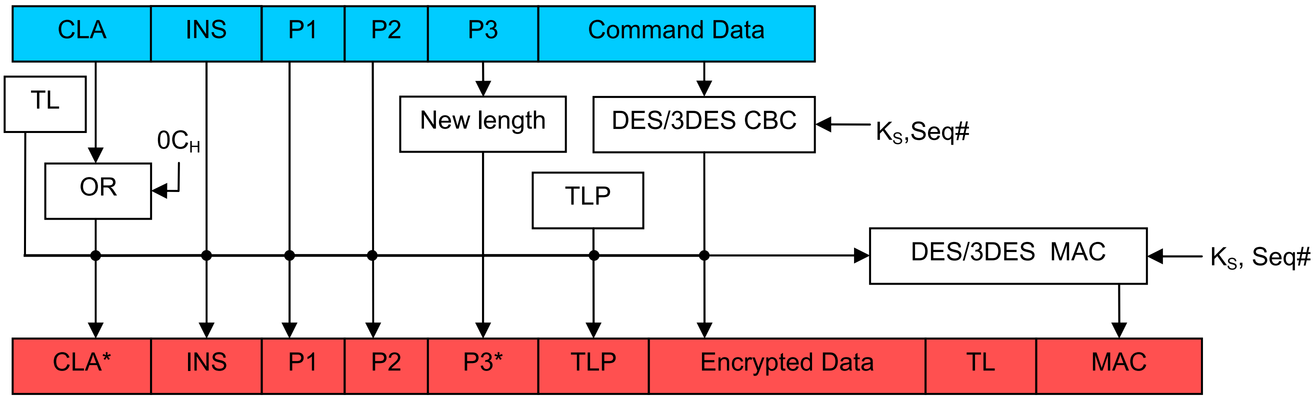


Figure 9: Secure Messaging for Confidentiality ISO-IN command transformation

The following figure shows the response transformation:

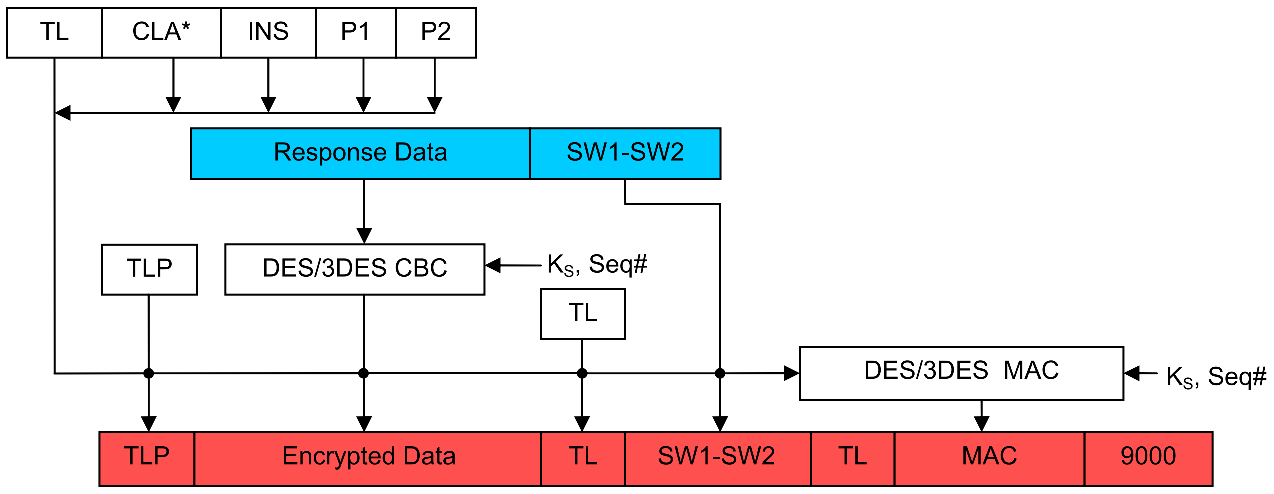


Figure 10: Secure Messaging for Confidentiality Response transformation

If there is no response data, that field does not appear in the response output.

The components necessary to compute secure messaging are explained in the sections below. The terms TL and TLP are *Tag-Length* and *Tag-Length-Padding* respectively described in Section 5.6.7.

5.6.2. Secure Messaging Key

The key used in secure messaging is the Session Key computed in Section 5.3.2. The SM Key is either DES or 3DES depending on the size of the Session Key generated.

5.6.3. Sequence Number

The sequence number (seq#) *n* is used as the initial vector in the CBC calculation. The start of the sequence number is the increment of the random number of the LAST TWO bytes of the Get Challenge command padded with 6 NULL bytes. The response data is the increment of the command sequence number. The sequence number is used to prevent man-in-the-middle attack.

If a command is sent from the terminal to the card SM'ed with a sequence number *n*, and secure messaging is successful, the card will reply with a MAC computed with *n* + 1. The next secure messaging command to send will use the sequence number *n* + 2. When the sequence number reaches 00 00 00 00 00 00 FF FF_H, the next number would be 00 00 00 00 00 00 00 00_H.

5.6.4. Authentication and Integrity

The COS and terminal will use the SM key and DES / 3DES in CBC mode to compute the MAC. Note that this is different from the Retail-MAC of SM-MAC in Section 5.5.3. The notation is as follows:

SIGN_CBC (data to sign, Initial Vector = seq#)

Only the first 4 bytes of the resultant MAC are used for the command and response data. When a SM command is sent to the card, the card will first verify the MAC_{cmd} with the command and command data. Only if the MAC_{cmd} is verified, will the COS execute the command. This will ensure that the data is genuinely from the terminal.

If the MAC_{cmd} verification failed (status bytes = 6988_H), the sequence number *n* would have been incremented. The next command should use *n* + 1. In the case that the sequence number is out of synchronization between the terminal and card, a new session key can be reestablished.

5.6.5. Confidentiality

The COS and terminal will use the SM key and DES / 3DES in CBC mode to compute the encryption. The notation is as follows:

ENC_CBC (plaintext to encrypt, Initial Vector = seq#)

After verifying the MAC$_{cmd}$ the COS will decrypt the command data encrypted by the terminal. This command data will be written to the card based on the write binary or record command.

5.6.6. Padding

DES and 3DES algorithms take input block sizes of 8. Therefore appropriate padding is necessary. If the data to sign or encrypt is not in a multiple of 8, a 80$_H$ byte is appended followed by 0 to 6 00$_H$ to make the data to sign to be a multiple of 8 bytes. If data to sign or encrypt is a multiple of 8, then no padding is applied.

5.6.7. Secure Messaging Data Object

Secure Messaging Data Objects (SMDOs) are necessary to describe the data elements fully when transferring in SM-ENC mode. The following SMDOs are supported in ACOS6 SM-ENC:

Tag	Length	Description
87$_H$	Var.	Padding-content indicator byte followed by cryptogram
89$_H$	4	Command header (CLA* INS P1 P2)
8E$_H$	4	Cryptographic checksum
97$_H$	1	Original P3 in an ISO-out command
99$_H$	2	Processing status bytes (SW1-SW2) of the command.

Table 22: Secure Messaging for Confidentiality Data Objects

The exact usage of these SMDOs is stated in the next section with the possible command and response data pair for SM.

5.6.8. Secure Messaging Semantics

Card commands are basically input and output commands– called ISO-IN and ISO-OUT. Some commands have both input and outputs and it needs to call GET RESPONSE to retrieve the outputs. This is called an ISO-IN-OUT command. In order to send out a secure messaging command, the normal APDU of Section 5.6.1 will be modified with SMDOs. The following two sections show how those modifications are to be done.

5.6.8.1. ISO-IN

In commands such as Write Record and Binary, the commands are extended with SMDOs as follows:

	CLA*	INS	P1	P2	P3*	87$_H$	L$_{87}$	Pi	Encrypted Data	8E$_H$	04$_H$	MAC$_{cmd}$
Bytes	1	1	1	1	1	1	1	1	n*	1	1	4

CLA* = CLA OR 0C$_H$
P3* = 3 + n* + 2 + 4
L$_{87}$ = n* + 1 (Length of Tag 87$_H$)
Pi = Padding indicator: 00$_H$ – No padding is used in encrypted data
01-07$_H$ – Number of padding bytes used in encrypted data
n* = P3 + length (padding) = P3 + Pi
Encrypted Data = ENC_CBC (<Command Data> padding, ++Seq#))
MAC$_{cmd}$ = SIGN_CBC (<89$_H$ 04$_H$ CLA* INS P1 P2> <87$_H$ L$_{87}$ Pi Encrypted Data> padding, Seq#)

Since the maximum of P3* must be less than 255, with the MAC, padding and the SMDO, the original P3 must be less than or equal to 240 bytes. Note that in the MAC calculation, the sequence number is not pre-incremented. This is because the encryption and the MAC will use the same sequence number with the encryption to be performed first.

If the command accepts secure messaging, the key has been established and the MAC_{cmd} is correct, the response will be $610C_H$. Note that $610C_H$ may not actually mean that the command is successful. It merely means that the Secure Messaging is successful. A subsequent call to GET RESPONSE will yield the actual status bytes stating success or error. The GET RESPONSE must have P3 = $0C_H$ exactly. Otherwise $6C0C_H$ will be replied without SM.

The response to GET REPSONSE is as follows:

	99_H	02_H	SW1-SW2	$8E_H$	04_H	MAC_{rsp}	9000_H
Bytes	1	1	2	1	1	4	2

MAC_{rsp} = SIGN_CBC (<89 04 CLA* INS P1 P2> <99 02 SW1-SW2> padding, ++Seq#)

The field SW1-SW2 is the actual status bytes returned for the command. The last status bytes 9000_H states that the GET RESPONSE is successful.

5.6.8.2. ISO-OUT

The ISO-out commands effectively become an ISO-in command when the SM field is set.

	CLA*	INS	P1	P2	P3*	97_H	01_H	P3	$8E_H$	04_H	MAC_{cmd}
Bytes	1	1	1	1	1	1	1	1	1	1	4

CLA* = CLA OR $0C_H$
P3* = 9
MAC_{cmd} = SIGN_CBC (<89_H 04_H CLA* INS P1 P2> <97_H 01_H P3> padding, ++Seq#)

If the command accepts secure messaging, the key has been established and the MAC_{cmd} is correct, the response will be $61xx_H$. Same as ISO-in, the status bytes of $61xx_H$ only means that SM on the command is successful. The actual success of the overall command will depend on the SW1-SW2 data object when a subsequent GET RESPONSE is called with P3 = xx, where xx can be 15-FD_H.

Note the get response must be called with P3 = xx exactly, else $6Cxx_H$ will be returned. This is to prevent man in the middle attack. The original data out *n* must be less than or equal to 240 bytes. Else, error status 6700_H will be returned.

The response is as follows:

	87_H	L_{87}	Pi	Encrypted data	99_H	02_H	SW1-SW2	$8E_H$	04_H	MAC_{rsp}	9000_H
Bytes	1	1	1	n*	1	1	2	1	1	4	2

L_{87} = n* + 1 (Length of Tag 87_H)
Pi = Padding indicator: 00_H – No padding is used in encrypted data
$01\text{-}07_H$ – Number of padding bytes used in encrypted data
Encrypted Data = ENC_CBC (<Response Data> padding, ++Seq#))
n* = P3 + length (padding) = P3 + Pi
MAC_{rsp} = SIGN_CBC (<89_H 04_H CLA* INS P1 P2> <87_H L_{87} Pi Encrypted Data> <99_H 02_H SW1-SW2> padding, Seq #)

Note that in the MAC calculation, the sequence number is not pre-incremented. This is because the encryption and the MAC will use the same sequence number with the encryption to be performed first.

5.6.8.3. ISO-IN-OUT

In commands such as INQUIRE ACCOUNT and DEBIT, the commands are extended with SMDOs as follows:

	CLA*	INS	P1	P2	P3*	87_H	L_{87}	Pi	Encrypted Data	$8E_H$	04_H	MAC_{cmd}
Bytes	1	1	1	1	1	1	1	1	n*	1	1	4

CLA* = CLA OR $0C_H$
P3* = 3 + n* + 2 + 4
L_{87} = n* + 1 (Length of Tag 87_H)
Pi = Padding indicator: 00_H – No padding is used in encrypted data
$01\text{-}07_H$ – Number of padding bytes used in encrypted data
n* = P3 + length (padding) = P3 + Pi
Encrypted Data = ENC_CBC (<Command Data> padding, ++Seq#))
MAC_{cmd} = SIGN_CBC (<89_H 04_H CLA* INS P1 P2> <87_H L_{87} Pi Encrypted Data> padding, Seq#)

Since the maximum of P3* must be less than 255, with the MAC, padding and the SMDO, the original P3 must be less than or equal to 240 bytes. Note that in the MAC calculation, the sequence number is not pre-incremented. This is because the encryption and the MAC will use the same sequence number with the encryption to be performed first.

If the command accepts secure messaging, the key has been established and the MAC_{cmd} is correct, the response will be $61xx_H$. Same as ISO-in, the status bytes of $61xx_H$ only means that SM on the command is successful. The actual success of the overall command will depend on the SW1-SW2 data object when a subsequent GET RESPONSE is called with P3 = xx, where xx can be $15\text{-}FD_H$.

Note the get response must be called with P3 = xx exactly, else $6Cxx_H$ will be returned. This is to prevent man in the middle attack. The original data out *n* must be less than or equal to 240 bytes. Else, error status 6700_H will be returned.

The response is as follows:

	87_H	L_{87}	Pi	Encrypted data	99_H	02_H	SW1-SW2	$8E_H$	04_H	MAC_{rsp}	9000_H
Bytes	1	1	1	n*	1	1	2	1	1	4	2

L_{87} = n* + 1 (Length of Tag 87_H)
Pi = Padding indicator: 00_H – No padding is used in encrypted data
$01\text{-}07_H$ – Number of padding bytes used in encrypted data
Encrypted Data = ENC_CBC (<Response Data> padding, ++Seq#))
n* = P3 + length (padding) = P3 + Pi
MAC_{rsp} = SIGN_CBC (<89_H 04_H CLA* INS P1 P2> <87_H L_{87} Pi Encrypted Data> <99_H 02_H SW1-SW2> padding, Seq #)

Note that in the MAC calculation, the sequence number is not pre-incremented. This is because the encryption and the MAC will use the same sequence number with the encryption to be performed first.

5.6.9. SM specific Response Status Bytes

The following table lists the specific SM status bytes returned by the card:

SW1	SW2	Meaning
61	xx	SM successful, call Get Response to retrieve xx bytes.
67	00	The original P3 is greater than 240 bytes. SMDO overflow.
68	82	Secure messaging not allowed.
69	82	Condition of use not satisfied – Secure Messaging required but not present.
69	85	The Session Key has not been established.
69	87	Expected secure messaging data objects missing.
69	88	The MAC_{cmd} does not match the data.
6C	xx	Get Repsonse with xx byte as P3 is expected. Please re-issue Get Response with P3=xx.

Table 23: Secure Messaging specific return codes

5.7. Interaction between Security Conditions and Internal Security EFs

At first glance, the previous sections may seem unclear as to how the DF or working EFs interacts with different internal EFs (SE File, Key File, PIN File) to provide security for the files in question. Figure 11: provide an example of how all these files work together.

In this example, the current DF has an SE file of F0 00_H. Inside this SE file, all the security environments are defined for the current DF. SE ID #1 defines the security conditions that must be satisfied before the current DF can execute a create child EF/DF or delete child. SE ID #1 contains an AT template for both external authentication and user verification with key reference of 01_H. In the DF's key file and PIN file the record of 01_H will define the key and PIN that must be satisfied before any create or delete file command can be called in this DF.

In the Working EF, the SAC in the file header has update security condition set with SE ID #3. When update record/binary command is called on this file, the card will check the SE ID number 3 and find the Authentication Template. It will then see that User Verification is needed using PIN ID of 02_H. Then, it will check if the PIN ID of 02_H in the PIN file has been verified beforehand.

Current DF/MF:

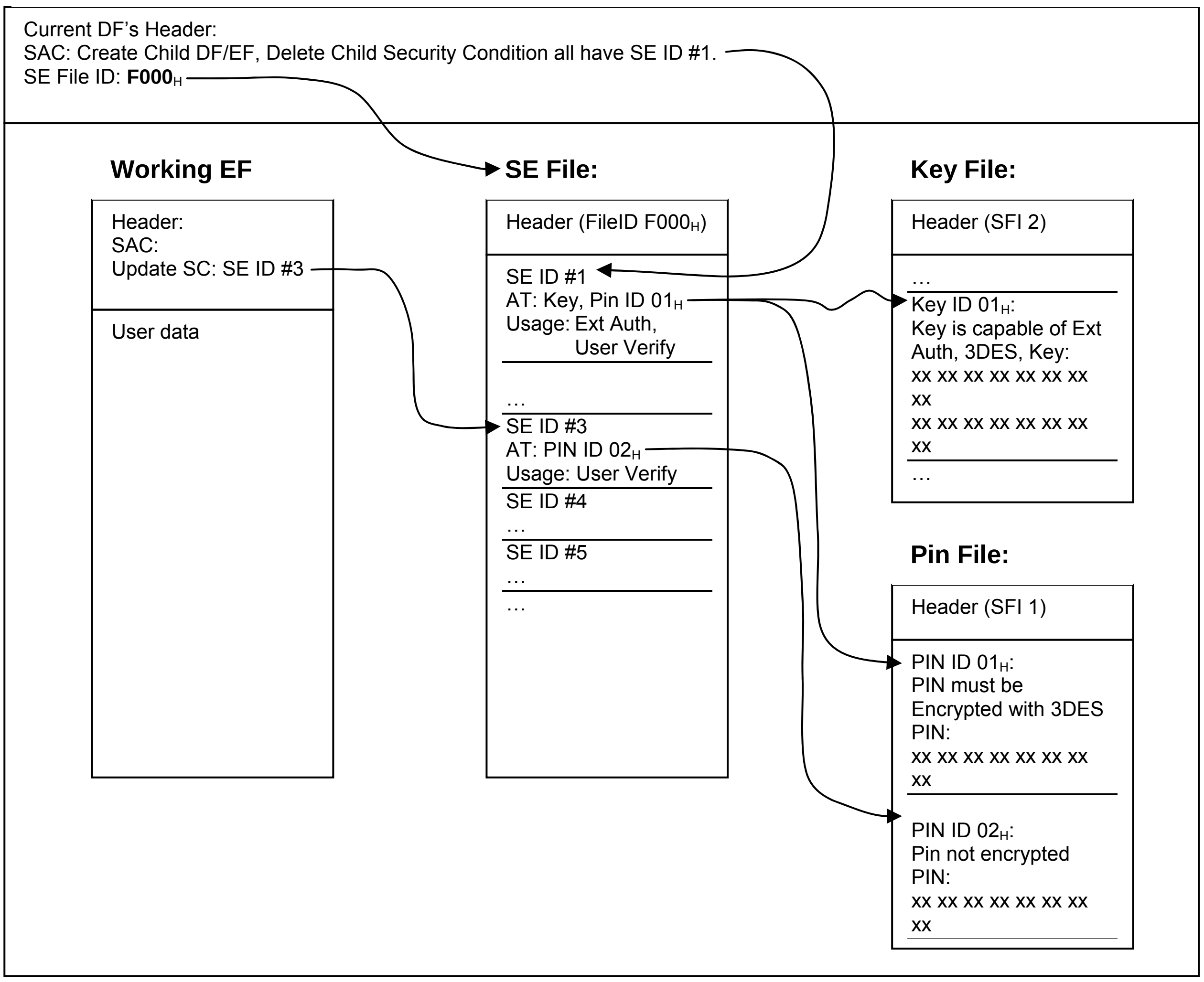


Figure 11: Relationship between working EFs and internal security files

5.8. Encrypted Code Operations

Depending on the setting of the PIN records in EF1 (Section 4.3.1), a code (or a PIN) may be submitted encrypted using the session key generated by mutual authentication. This section describes how to submit and change codes using encryption.

5.8.1. Submit Encrypted Code

If the setting in the PIN Identifier Byte of the PIN code to be submitted has b6 set, code submission must be encrypted. Depending on b5 of the PIN Identifier Byte, the PIN submission must be DES or 3DES. If b5 is set to 3DES, then the session key K_S must be 16-byte 3DES key or else, the PIN submission will not be possible. If b5 is set to DES and the session key K_S is 16-byte, the COS will only use the first 8-byte of the session key K_{SL} for PIN submission.

To submit the code C_i that has identifier *i*, the following procedure is executed after a session key K_S has been established:

Terminal		ACOS6-SAM
Compute R = ENC (C_i, K_S)		
Submit encrypted code 00 20 00 *i* 08 *R*	➔	
		Check if code identifier *i* and session key K_S is valid.
		Verify that R = ENC (C_i, K_S)
	🡐	Returns 9000 if successful

Figure 12: Submit encrypted code procedure

In the procedure, the encryption, ENC, is either DES or 3DES depending on b5 of the PIN Identifier Byte.

5.8.2. Change Encrypted Code

The PIN code can be changed during the activated state of the card if b7 of PIN Identifier Byte is set. If the PIN code requires encryption (b6 of PIN Identifier Byte is set), the following procedure is performed:

Terminal		ACOS6-SAM
Submit Current PIN		Same procedure as Section 5.8.1.
Get new PIN		
Compute R = DEC (new PIN, K_S)		
Change PIN 00 24 00 *i* 08 *R*	➔	
		Check if code identifier *i*, its flags, and session key K_S is valid.
		Compute C_i = ENC (R, K_S)
	🡐	Set new code C_i in PIN file with identifier *i*. Returns 9000 if successful

Figure 13: Change encrypted code procedure

In the procedure, the encryption, ENC, and decryption, DEC, is either DES or 3DES depending on b5 of the PIN Identifier Byte.

5.9. Key Injection

Key injection can be used to securely load a key or diversified key from an ACOS6-SAM card into a target ACOS6-SAM or client ACOS6 card. For the purpose of key injection, we shall refer to the ACOS6-SAM with the key to inject the "*source SAM*" and the ACOS6/ACOS6-SAM to receive the key the "*target SAM*".

This function allows for a master and subordinate SAM relationships and the subordinate SAMs can perform different specific operations.

The target SAM cards uses the Set Key command and the source SAM will use the Get Key command to perform key injection. .

The keys to be injected will be encrypted and MAC'ed with a session key established previously and with the nonce initial vector. The session key is generated by the mutual authentication as in Section 9.2. This means that a shared set of internal authenticate and external authenticate keys must already reside in both source and target SAMs. This can be done during pre-personalization of the cards in a secured facility.

Each time a key injection is done, a set of random numbers will be generated in both source and target SAM to be used as the initial vector for the encrypted key and MAC. This ensures the encrypted keys cannot be replayed.

The encryption and MAC are performed by the source SAM's Get Key command is shown in the figure below. The resultant encrypted data and MAC are sent into the target SAM's Set Key command.

Note: The key injection feature is available for ACOS6-SAM revision 4.02 and ACOS6 revision 3.02 onwards.

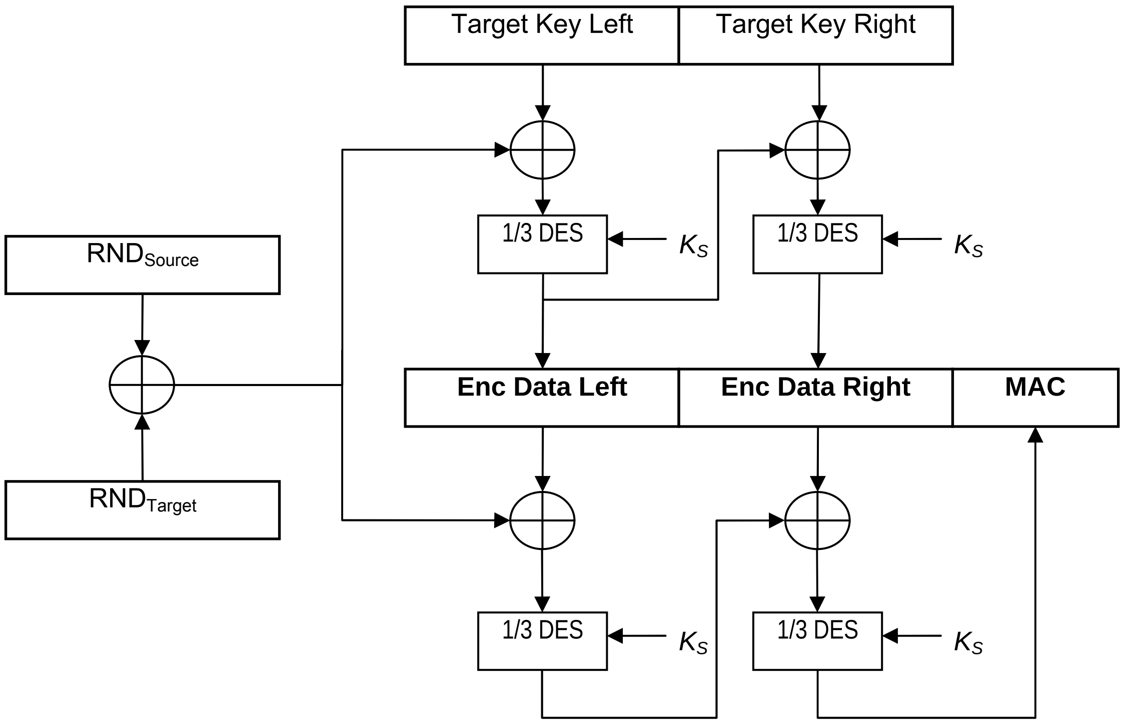


Figure 14: Key injection encryption computation.

6.0. Commands

This section contains the command set of this card excluding the SAM specific commands. Most of these commands are defined in ISO7816 part 4. The following contain a summary table of all commands.

Command	Instruction	Description
Create File	$E0_H$	Create MF/DF/EF according to supplied parameters
Select File	$A4_H$	Select MF/DF/EF files
Read Binary	$B0_H$	Read data from referenced binary file.
Update Binary	$D6_H$	Update data from referenced binary file.
Read Record	$B2_H$	Read record data from referenced record file.
Update Record / Write Record	$D2_H$ / DC_H	Update record data from referenced record file.
Append Record	$E2_H$	Append a record into the linear variable record file.
Activate File / Card	44_H	Activate a file which ensure all security attributes of the file. Activating the card sets card from personalization state to user state.
Deactivate File / Card	04_H	Deactivate the referenced file to disable access. Deactivating the card sets card from user state back to personalization state.
Terminate DF	$E6_H$	Set the target DF to TERMINATED state.
Terminate EF	$E8_H$	Set the target EF to TERMINATED state.
Delete File	$E4_H$	Delete the last created file.
Get Challenge	84_H	Get card random challenge data for authentication purpose.
Mutual / External Authentication	86_H	Perform mutual authentication or short key external authentication.
Verify	20_H	Verify a code
Change Code	24_H	Change a code
Get Card Info	14_H	Get information about card serial number
Get Response	$C0_H$	Get result of a previous command.
Clear Card	30_H	Clear all EEPROM data and return the card to the pre-perso state.
Set Key	DA_H	Inject key from ACOS6-SAM or terminal.
Generate Key	88_H	Generate a diversified key to load into a client card.
Diversify (or Load) Key Data	72_H	Diversify or load a key internally for ciphering operations.
Encrypt	74_H	Encrypt data using session, diversified or loaded keys.
Decrypt	76_H	Decrypt data using session, diversified or loaded keys.
Prepare Authentication	78_H	Perform first half of ACOS authentication procedure.
Verify Authentication	$7A_H$	Perform second half of ACOS authentication procedure.
Verify ACOS Inquire Account	$7C_H$	Verify ACOS2/3/6 inquire account purse command.
Prepare ACOS Account Transaction	$7E_H$	Create a debit/credit command for ACOS card.
Verify Debit Certificate	70_H	Verify ACOS3/6's debit certificate.

Get Key	CA_H	Export encrypted key to inject into an ACOS6 client card or another ACOS6-SAM card.

Note: Some instruction codes INS are the same. These are distinguished by the class CLA byte.

Table 24: Command table summary

6.1. Create File

Create a MF/DF/EF based on a set of attributes.

CLA	00 - Clear Mode 04 - SM Mode
INS	E0
P1	00
P2	00
P3	Length of Data
Data	FCP TLV of File to be created

The FCP TLV has tag of 0x62. Its Length is size of the template, and must be equal to (P3 – 2). The template has one or more of the following encapsulated TLV's:

Tag	Length	Value	MF	DF	Transparent	Linear Fixed	Linear Variable	Cyclic	Key EF	Purse EF	Remarks
80	02	Size (in bytes) of the Transparent EF			O						If not specified, default is 0x0000
82	01	File Descriptor Byte (FDB)	M	M	M	M	M	M	M	M	Please refer to Section 4.2.1 for valid values
	02	FDB || DCB	O	O	O	O	O	O	O	O	If DCB is not specified, default is 0x00
	05	FDB || DCB || 00 || MRL || NOR				O	O	O	O	O	If MRL / NOR is not specified, default is 0x00
	06	FDB || DCB || 00 || MRL || 00 || NOR				O	O	O	O	O	If MRL / NOR is not specified, default is 0x00
83	02	File ID	M	M	M	M	M	M	M	M	Please refer to Section 4.2.3 for valid values
84	<= 10h	DF Long Name	O	O							
88	01	Short File ID	O	O	O	O	O	O	O	O	If not specified, the default is the 5 LSB of the File ID
8A	01	Life Cycle State Integer	O	O	O	O	O	O	O	O	If not specified, the default is 0x01. Refer to section 2.1.6 for valid values.
8C	<= 08	Security Attributes Compact	O	O	O	O	O	O	O	O	Please refer to Section 5.1.1 for more details
AB	<= 20h	Security Attributes Extended	O	O							Please refer to Section 5.1.2 for more details
8D	02	SE File ID	O	O							Please refer to Section 4.2.14 for more details
87	02	FCI File ID	O	O							Please refer to Section 4.2.15 for more details

M – Mandatory
O – Optional
Blank – encapsulated TLV will be ignored

The mandatory fields are: (1) File ID and (2) FDB. The newly created file will then be the Currently Selected EF/DF.

If duplicate tags are sent to the command, the latter one will be used.

Valid FDB values are: 3F (MF), 01 (Transparent EF), 02 (Linear Fixed EF), 04 (Linear Variable EF), 06 (Cyclic EF), 0x0C (Internal LV EF, or KEY EF), and 0x0E (Internal Cyclic EF, or Purse EF).

File ID's cannot be: 0xFFFF, 0x0000 and 0x3FFF.

Valid LCSI are: 01 (Creation); 03 (Initialization); 04 or 06 (Deactivated); 05 or 07 (Activated). LCSI having value >= 08 are considered Terminated.

The MF's file ID should always be 0x3F00, and should always be the 1st created file. You can create as many DF / EF files, and as many DF levels, as long as the card memory space holds. There cannot be duplicate File ID's / DF Names under a DF.

Please refer to Sections 10.0 and 11.0 of this document for examples on using this command.

Specific Response Status Bytes:

SW1-SW2	Description
6A86	Incorrect P1 / P2
6700	Wrong P3, must be consistent with the Length of the FCP TLV
6A80	Wrong FCP Tag - should be 0x62
	FCP contains invalid TLV's, unrecognized tags, or wrong length's in TLV's
	Creating MF, but MF already exists
	File ID is 3F00 but FDB is not MF or MF already exists
	Invalid File ID, FDB, SFI, LCSI, etc.
6283	Current DF is terminated/ deactivated
6982	Security condition not satisfied
6A89	File ID / DF Name already exists
6A84	Not enough free space in card to create file

6.2. Select File

Select a target MF/DF/EF based on File ID or DF Name.

CLA	00 - CLEAR mode 80 - ACOS2 mode
INS	A4
P1	04 - if Data contains DF name and in CLEAR mode, else 00
P2	00
P3	00 - select MF, 02 - Data contains File ID (or in ACOS2 mode), else length of DF Name
Data	File ID or DF Name

Search Sequence for Target File ID is: current DF -> current DF's children -> current DF’s parent -> current DF’s siblings -> MF -> MF’s children.

Search Sequence for Target DF Name is: current DF -> current DF’s children -> current DF’s parent

On success, in ACOS6-SAM mode, SELECT FILE will return SW1-SW2 = 61XX. You can retrieve XX bytes (via GET RESPONSE command with P3 = XX) to get the FCI template of the selected file. The template complies with the TLV table defined in 4.1.

On success, in ACOS2 mode, SELECT FILE will return SW1-SW2 = 91nn, where nn the file index of the selected file. SW1-SW2 is 9000 if the selected file is a system ACOS2 file.

In ACOS2 mode, file searching only applies to the MF level.

Specific Response Status Bytes:

SW1-SW2	Description
6283	Target file is blocked, but is selected
6982	Target file has wrong checksum in header
6986	No MF found in card
6A82	File not found
6A86	Invalid P1/P2
6700	Wrong P3, P3 not compatible to P1/P2
9000	System File selected successfully under ACOS2 mode.
91nn	User File selected successfully under ACOS2 mode.
61nn	File selected successfully under ACOS6-SAM mode. Issue GET RESPONSE with P3 =NN in order to retrieve the file’s FCI template

6.3. Read Binary

Reads out the contents of a Transparent File, given the file offset.

CLA	00 - Clear mode 04 - SM mode
INS	B0
P1	If MSb = 1, P1 holds SFI in 5 LSb, else P1 holds high offset
P2	Low offset
P3	Bytes to Read

If MF does not exist, READ BINARY allows you to directly read the EEPROM's contents. P1-P2 holds the physical address while P3 holds the number of bytes to read (please refer to Section 3.2 for valid EEPROM physical addresses).

If MF exists, this command follows READ BINARY command specified in ISO7816 part 4.

Specific Response Status Bytes:

SW1-SW2	Description
6282	Invalid offset
6283	Current DF is blocked or target EF is blocked
6700	Incorrect P3, length exceeds file size limit
6981	Wrong file type; target file must be TRANSPARENT EF
6982	Target file's header block has wrong checksum, or security condition not satisfied
6986	No EF selected
6A82	SFI not found
6B00	Invalid P1/P2, invalid SFI
6Cnn	Wrong P3; NN = maximum bytes available in file to read
6F00	Invalid physical address (when directly accessing EEPROM)

6.4. Update Binary

Writes data to a Transparent File, given the offset.

CLA	00 - Clear mode 04 - SM mode
INS	D6
P1	If MSb = 1, P1's 5 LSb holds SFI, else P1 holds high start offset
P2	Low start offset
P3	Number of Bytes to Update
Data	Bytes to Update

If MF does not exist, UPDATE BINARY allows you to directly write the EEPROM contents. P1-P2 holds the starting physical address of the card (please refer to section 3.1 for valid EEPROM physical addresses).

If MF exists, this command follows UPDATE BINARY command specified in ISO7816 part 4.

Specific Response Status Bytes:

SW1-SW2	Description
6282	Invalid offset
6283	Current DF is blocked, or target EF is blocked
6700	Incorrect P3, length exceeds file size limit
6981	Wrong file type; target file must be TRANSPARENT EF
6982	Target file's header block has wrong checksum, or security condition not satisfied
6986	No EF selected
6A82	SFI not found
6B00	Invalid P1/P2, or invalid SFI
6Cnn	Wrong P3; NN = maximum bytes available in file to update
6F00	Invalid physical address (when directly accessing the EEPROM), or write failure

6.5. Read Record

Reads out the contents of a record block from a Record-based EF.

CLA	00 - Clear mode;
	04 - SM mode;
	80 - ACOS2 mode

INS	B2
P1	Record number (in ACOS2 mode) Record number (in ACOS6-SAM mode based on P2)
P2	If 5 MSb <> 00000, it is SFI; 3 LSb: see below
P3	Bytes to Read

In ACOS6-SAM mode, the last 3 bits of P2:

0 : reference 1st record
1 : reference last record
2 : reference next record
3 : reference previous record
4 : reference record indexed by P1
others : RFU – 000_b

If the file's MRL or NOR field is zero, the COS will return 6A83 status bytes (record not found).

For Linear EF's, it is possible to have P3 < the EF's MRL.

Specific Response Status Bytes:

SW1-SW2	Description
6283	Current DF is blocked, or Target EF is blocked
6700	Incorrect P3
6981	Wrong file type; target file must be RECORD-BASED EF
6982	Target file's header block has wrong checksum, or security condition not satisfied
6986	No DF selected, or no EF selected
6A82	SFI not found
6A83	Record not found, invalid record reference
6A86 6B00	Invalid P1/P2 in ACOS2 mode Invalid P1/P2, invalid SFI in ACOS6-SAM mode
6Cnn	Wrong P3, NN = maximum bytes available in file to read

6.6. Update Record

Writes contents of a record block in a Record-based EF.

CLA	00 - Clear mode 04 - SM mode 80 - ACOS2 mode
INS	DC
P1	Record number (in ACOS2 mode) Record number (in ACOS6-SAM mode based on P2)
P2	If 5 MSb <> 00000, it is SFI; 3 LSb: see below
P3	Number of Bytes to Write
Data	Bytes to Write

Last 3 bits of P2:

0 : reference 1st record
1 : reference last record
2 : reference next record
3 : reference previous record
4 : reference record indexed by P1

For Linear EF, P3 can be < the file's MRL.

Specific Response Status Bytes:

SW1-SW2	Description
6283	Current DF is blocked, or Target EF is blocked
6700	Incorrect P3
6981	Wrong file type; target file must be RECORD-Based EF
6982	Target file's header block has wrong checksum, security condition not satisfied
6986	No DF selected, no EF selected
6A82	SFI not found
6A83	Record not found, invalid record reference
6A86 6B00	Invalid P1/P2 in ACOS2 mode Invalid P1/P2, invalid SFI in ACOS6-SAM mode
6Cnn	Wrong P3, NN = maximum bytes to write to record

6.7. Write Record

This command does exactly the same thing as UPDATE RECORD. It is included for ACOS2 compatibility.

CLA	00 - Clear mode 04 - SM mode 80 - ACOS2 mode
INS	D2
P1	Record number (in ACOS2 mode) Record number (in ACOS6-SAM mode based on P2)
P2	If 5 MSb <> 00000, it is SFI; 3 LSb: see below
P3	Number of Bytes to Write
Data	Bytes to Write

6.8. Append Record

Adds a record at the end of a Linear Variable EF.

CLA	00 - Clear Mode 04 - SM Mode
INS	E2
P1	00
P2	00
P3	Length of Data
Data	Data to be appended

This command applies only to Linear Variable EF. The new record will be written to the 1st record whose 1st byte is 0xFF (a record whose 1st byte is 0xFF is considered ‘empty’, and therefore is at the ‘end’ of the file). P3 can be less than the file’s MRL, in such case; 0xFF will be padded to the new record.

Specific Response Status Bytes:

SW1-SW2	Description
6283	Target file / current DF is blocked
6981	Target file is not Linear Variable EF
6982	Target file has wrong checksum in header
6982	Security condition not satisfied
6A83	Record not found
6986	No file selected
6A84	No more empty record in EF
6B00	Wrong P1 / P2
6700	Invalid P3

6.9. Activate File / Card

This command will activate the target file. Once activated, the file's security settings will take effect.

CLA	00 - Clear Mode 04 - SM Mode
INS	44
P1	00 – activate File 01 – activate Card
P2	00
P3	02 – Activate the File whose ID is referenced by the command data 00 – Activate the currently selected EF or DF or the card
Data	File ID

If P1 = 0x00, this command activates the file referenced in command data. User can activate the following files: (1) current DF and (2) child file of the current DF.

If P1 = 0x01, this command activates the card by setting the card life cycle fuse to 0x00 even when the card is in pre-personalization state.

Specific Response Status Bytes:

SW1-SW2	Description
6400	Target file is terminated
6982	Target file has wrong checksum, or security condition not satisfied
6A82	File referenced not found
6986	No file selected
6A86	Invalid P1 / P2
6700	Invalid P3, must be 2
6F00	EEPROM failure

6.10. Deactivate File / Card

This command will invalidate or deactivate the target file. Once a file is deactivated, all commands (except ACTIVATE FILE) to the file will be rejected.

CLA	00 - Clear Mode 04 - SM Mode
INS	04
P1	00 – activate File 01 – activate Card
P2	00
P3	02 – Deactivate the File whose ID is referenced by the command data. 00 – Deactivate the currently selected EF or DF, or the card.
Data	File ID

If P1 = 0x00, this command deactivates the file referenced in DATA. User can deactivate the following files: (1) current DF and (2) child file of the current DF.

If P1 = 0x01, this command resets the card life cycle fuse to 0xFF. This command can only be called in MF level and bit 5 - Deactivate Card Enable Flag of the Card Header Block (Section 3.2) must be set. If access condition is required for this command, a SAE can be set at the MF level to allow access after PIN verification or key authentication.

Specific Response Status Bytes:

SW1-SW2	Description
6400	Target file is terminated
6982	Target file has wrong checksum, or security condition not satisfied
6A82	File referenced not found
6986	No file selected
6A86	Invalid P1 / P2
6700	Invalid P3, must be 2
6F00	EEPROM failure

6.11. Terminate DF

This command will irreversibly send the target DF to TERMINATED state. In such case, all commands to the file will be rejected.

CLA	00 - Clear Mode 04 - SM Mode
INS	E6
P1	00
P2	00
P3	02 – Terminate the DF File whose ID is referenced by the DATA 00 – Terminate the currently selected DF
Data	File ID

This command terminates the DF file referenced in DATA. User can terminate the following files: (1) current DF and (2) child DF file of the current DF.

Specific Response Status Bytes:

SW1-SW2	Description
6400	Target file is terminated
6982	Target file has wrong checksum, security condition not satisfied
6A82	File referenced not found
6986	No file selected
6A86	Invalid P1 / P2
6700	Invalid P3, must be 2
6F00	Update failure
6981	File is not DF type

6.12. Terminate EF

This command will irreversibly send the target EF to TERMINATED state. In such case, all commands to the file will be rejected.

CLA	00 - Clear Mode 04 - SM Mode
INS	E8
P1	00
P2	00
P3	02 – Terminate the EF File whose ID is referenced by the DATA 00 – Terminate the currently selected EF
Data	File ID

This command terminates the EF file referenced in DATA. User can terminate the following files: (1) current EF and (2) child EF file of the current DF.

Specific Response Status Bytes:

SW1-SW2	Description
6400	Target file is terminated
6982	Target file has wrong checksum, or security condition not satisfied
6A82	File referenced not found
6986	No file selected
6A86	Invalid P1 / P2
6700	Invalid P3, must be 2
6F00	Update failure
6981	File is not EF type

6.13. Delete File

This command will remove the target file from the card's EEPROM memory.

CLA	00 – Clear Mode 04 - SM Mode
INS	E4
P1	00
P2	00
P3	02 – Delete the File whose ID is referenced by the DATA 00 – Delete the currently selected DF or EF
Data	File ID

Delete File will work if the target file is the last file created in EEPROM memory area. ACOS6-SAM will not allow deletion of a DF that has children.

Specific Response Status Bytes:

SW1-SW2	Description
6400	Target file is terminated
6982	Target file has wrong checksum, or security condition not satisfied
6A82	File referenced not found
6986	No file selected
6A86	Invalid P1 / P2
6700	Invalid P3, must be 2
6F00	EEPROM failure
6981	File is not EF type
6A80	Target DF file has children Target file is not the last file in memory

6.14. Get Challenge

This command generates a 4/8-byte Challenge Data, to be used for authentication purposes.

CLA	00 - ACOS6-SAM mode 80 - ACOS2 mode
INS	84
P1	00
P2	00
P3	04 / 08

A response data of 4-byte challenge is used for short-key external authentication while an 8-byte is used for mutual authentication.

Specific Response Status Bytes:

SW1-SW2	Description
6700	Incorrect P3
6A86	Wrong P1 / P2

6.15. Mutual / External Authentication

This command authenticates the referenced key(s) of the currently selected DF. There are two types of authentication in ACOS6: mutual authentication and short key external authentication.

Mutual authentication proves both the identity of the terminal to the card (external authenticate) and card to the terminal (internal authenticate). The short key external authenticate proves only the terminal to the card using an input key which is optimal for one-time password for card access application.

Appropriate access rights are gained if successful based on the SE file and file access conditions.

CLA	00 – ACOS6-SAM mode 80 – ACOS2 mode
INS	82
P1	Key index of Internal Key (key type bit 2 must be set); if MSb=1, use local EF2 else use global EF2 P1= 00h if short key external authenticate is used.
P2	Key index of External Key (key type bit 3 or 0 must be set); if MSb=1, use local EF2 else use global EF2
P3	10 or 04h
Data	If P3=10h, command data is xDES (RNDc, Kt) || RNDt If P3=04h and P1=00h, command data is the feft 4 byte of xDES (RNDc, Kt) xDES may be Single DES or Triple DES; depending on the attributes of the keys used

If P3=04h, on success, COS will return 0x9000.

If P3=10h, on success, COS will return 0x6108. The result is: xDES (RNDt. Ks).

In ACOS2 mode, P1 and P2 should be 0x00. In such case, Card Key is the 1st record in global EF2 and Terminal Key is the 2nd record in global EF2.

Triple DES will be used if both Keys (referenced by P1 and P2) has ALGO field of 0x00. In this case, if either key's length is less than 16, 0xFF will be padded.

Single DES will be used if either Key (referenced by P1 and P2) has ALGO field not equal to 0x00. In this case, if either key's length is > 8, it will be truncated to 8 (i.e., by using the first 8 bytes only).

Please see Section 5.3 for a more detailed description of the process.

Specific Response Status Bytes:

SW1-SW2	Description
6A87	KEY(s) referenced in P1/P2 not capable of Internal/External authentication
63Cn	Wrong Crypto Data, Operation Fail, n tries left for KEY
6700	Incorrect P3, must be 16
6983	Terminal KEY is locked or Card Key is locked
6985	GET CHALLENGE command not previously called
6A86	Invalid P1 / P2 in ACOS2 mode
6986	No DF selected
6283	Current DF is blocked, EF2 is blocked
6A88	EF2 not found
6A83	Key not found in EF2, key has invalid length
6981	Invalid EF2 (FDB, MRL, etc not consistent)
6108	Authentication OK

6.16. Verify

This command is used to submit a PIN code to gain access rights.

CLA	00 - ACOS6-SAM mode 80 - ACOS2 mode
INS	20
P1	If ACOS2 mode: PIN ID: if MSb = 1, use Local PIN, else use Global PIN
P2	If ACOS6-SAM mode: PIN ID: if MSb = 1, use Local PIN, else use Global PIN
P3	Length of PIN
Data	PIN

If the PIN is encrypted, its length should always be 8.

In ACOS2 mode, only Global PIN applies, and P3 should be 8.

Access rights achieved will be invalidated when a new DF is selected.

Specific Response Status Bytes:

SW1-SW2	Description
6283	Current DF is blocked; EF1 is blocked
63Cn	Verify fail, n tries remaining
6700	Incorrect P3, does not match PIN length, must be <= 32
6981	EF1 has wrong FDB, EF1 is not a valid PIN file
6983	Referenced PIN is locked
6986	No DF selected
6A83	PIN ID referenced in P1 is not found in EF1
6A86	Invalid P1/P2
6A88	EF1 not found
6985	Session Key not valid, or not consistent with key DES mode (Session key is needed for encrypted PIN)

6.17. Change Code

This command allows you to change a PIN code in EF1.

CLA	00 - ACOS6-SAM mode 80 - ACOS2 mode
INS	24
P1	If ACOS2 mode: PIN ID: if MSb = 1, use Local PIN, else use Global PIN
P2	If ACOS6-SAM mode: PIN ID: if MSb = 1, use Local PIN, else use Global PIN
P3	20h or less, or 08 if PIN is encrypted or if in ACOS2 mode
Data	New PIN

This command will work only if you have successfully verified the PIN ID previously.

When PIN is encrypted, the new PIN in DATA must be decrypted with the Session Key. The length of encrypted PIN is always 8 bytes.

In ACOS2 mode, only Global PIN applies, and P3 should be 8.

Specific Response Status Bytes:

SW1-SW2	Description
6283	DF is blocked; EF1 is blocked
6700	Incorrect P3, does not match KEY length must be <= 32
6981	EF1 has wrong FDB, EF1 is not a valid PIN file
6983	Referenced PIN is locked
6986	No DF selected
6A83	Referenced PIN ID not found in EF1
6A86	Invalid P1/P2
6A88	EF1 not found
6985	Session Key not valid, not consistent with key DES mode
6966	PIN_ALT of the PIN is disabled
6982	PIN not verified previously

6.18. Get Card Info

This command returns card or file information of the ACOS6 card.

CLA	80
INS	14
P1	See options below
P2	
P3	

The following card information is available depending on the parameters of the command.

P1	P2	P3	Description
00	00	08	Returns card's unique serial number
01	00	00	Returns the number of files under the currently selected DF in SW1-SW2 = 90XX, where XX is the number of files.
02	xx	08	Returns file information of the P2th file in the DF. The 8 bytes are: {FDB, DCB, FILE ID, FILE ID, SIZE or MRL, SIZE or NOR, SFI, LCSI};
04	00	06	Card returns the 6-byte Card ID Number (refer to Section 3.2).
05	00	00	Returns the size of EEPROM in the status bytes SW1-SW2 = 90XX.
06	00	08	Returns the version number of the ACOS6 in the form of ACOS6 Revision XX YY ZZ (41 43 4F 53 06 XX YY ZZ $_H$)

Specific Response Status Bytes:

SW1-SW2	Description
6700	Incorrect P3
6A80	Wrong P1/P2, data not available

6.19. Get Response

This command returns information available in the card OS, with regards to the previous command.

CLA	00 - ACOS6-SAM mode 80 - ACOS2 mode
INS	C0
P1	0
P2	0
P3	Bytes to receive

Specific Response Status Bytes:

SW1-SW2	Description
6A86	Wrong P1 / P2
6Cnn	Incorrect P3, P3 must be nn
6985	No data available

6.20. Clear Card

This command will set the card back to Pre-Perso State. It is available only if the card is in Perso State.

CLA	80
INS	30
P1	00
P2	00
P3	00

On success, the card's entire EEPROM memory will be erased (set to 0xFF). This command is only available if the card's header info field: Card Life Cycle Status (address 0xEEC7) is not set (0xFF). This command has anti-tearing protection. That is, if the card loses power during the execution of this command, the card will continue execution the next time it is powered up. Hence, ensuring all data are cleared.

Specific Response Status Bytes:

SW1-SW2	Description
6F00	Command not available

6.21. Set Key

Set key allows secure key injection from the terminal or another ACOS6-SAM to a key file of the ACOS6. Using this ensures the keys to be injected is protected by encryption and message authentication codes.

If bit 7 - Key Injection Only Flag of the Card Header Block (Section 3.2) has been set and the key file has been activated, Set Key must be used for loading or changing keys in the card. Update Record command is not allowed.

Before this command is to be executed, a session key is already established with the mutual authentication procedure of Section 5.3.

CLA	80
INS	DA
P1	00
P2	Key ID
P3	0x1C
Data	RNDmsam + Encrypted Key Data + MAC

The security condition of Set Key is the Update/Set Key access condition of KEY File.

The Data is the same as output response data of GET KEY command of the source ACOS6-SAM card. Please see Section 5.9 for more information.

Specific Response Status Bytes:

SW1-SW2	Description
6985	GET CHALLENGE command not previously called or session key not ready
6283	Current DF is blocked, or Target EF is blocked
6986	No DF selected
6981	Wrong file type of Key file, it should be Internal Linear Variable File
6982	Target file's header block has wrong checksum, or security condition not satisfied
6A86	Invalid P1 or P2
6700	Incorrect P3
6A83	Target Key is not ready or Key Length less than 16
9000	Success

7.0. SAM Specific Commands

This section contains the SAM specific commands. For command flows of how to use these command please refer to Section 9.0.

7.1. Generate Key

This command is to generate a diversified key to load into the ACOS2/3/6 card or other cards from deviation data such as a client card serial number. This command is catered for client card issuance purposes.

CLA	80
INS	88
P1	00 – Generate 8-byte key from deviation data 01 – Return 16-byte diversified key from deviation data
P2	Key index of Master Key to generate Derived Key
P3	08 if P1=00 10 if P1=01
Data	Deviation data

The master key must be capable of performing internal authenticate in its key type attribute. It is essentially an encrypt (or internal authenticate) command. The command will decrement the usage counter by 1. This can be used to limit the number of ACOS3/6 cards that this SAM can issue.

Since Generate Key command is a sensitive command, it should have security restriction for its use. This can be achieved by a special SAE to a particular verification and/or authentication.

To generate 16-byte 3DES key, there are two methods:

Generate left half and right half separately: First issue this command with P1=00_H to generate the first 8 bytes of the 16-byte 3DES key. For the second part, bit-wise complement the plain text or serial number and issue this command again.

Generate left and right half in one command: Issue this command with P1=01_H. The return value would equal to the 16 byte 3DES diversified key.

The 16-byte 3DES key will be:

DERIVED KEY = ENC (DeviationData, MasterKey) || ENC (COMPL (DeviationData), MasterKey)

Specific Response Status Bytes:

SW1-SW2	Description
6986	No DF selected
6A86	Invalid P1 or P2
6700	Incorrect P3, must be 0x08
6A83	Referenced key record not found in EF2
6981	Invalid EF2 (record size, file type, etc.)
6A88	EF2 not found
6283	Current DF is blocked; EF2 is blocked
6982	Security condition not satisfied
6A87	Referenced Master Key is not capable of 3-DES encryption
6108	Command completed, issue GET REPONSE to get the result

7.2. Diversify (or Load) Key Data

This command prepares the SAM card to perform ciphering operations by diversifying and loading the key. It takes the deviation data (such as a client card's serial number) and CBC initial vector as command data input.

CLA	80
INS	72
P1	Target Key: 1 – Secret Code (S_C) 2 – Account Key (K_{ACCT}) 3 – Terminal Key 4 – Card Key 5 – Bulk Encryption Key (Not diversified) 6 – Initial vector
P2	Index of Master Key: Bit7: 1 = local key in current EF2; 0 = global key EF2 Bit6-Bit5: 00_b - RFU Bit4-Bit0: Key Index
P3	If P1 = 1-4, 6, P3 = 8 If P1 = 5, P3 = 0
Data	If P1 = 1-4 Client card's deviation data If P1 = 5, No command data. If P1 = 6, 8 byte DES/3DES CBC initial vector.

If P1 = 1 to 4, this command will generate a diversified Target Key by using this formula:

DERIVED KEY = ENC (DeviationData, MasterKey) || ENC (COMPL (DeviationData), MasterKey)

If subsequent commands use only single DES, the right half of DERIVED KEY would not be used.

If P1 = 5, the bulk encryption key will be loaded and it will not be diversified.

DERIVED KEY = MasterKey

The MasterKey in this case must have bulk encryption bit enabled in its Key Type attribute. This type of key is useful for using the SAM as an encryption engine. The data in P3 should be 0x00 or it will be ignored.

Note that the Master Keys or bulk encryption Target key must be a 3DES key since single DES is considered insufficient for most security applications. Depending on the application, the diversified key can be used for single DES operations.

If P1 = 6, the initial vector for CBC or MAC encryption is loaded. P2 value will be ignored. The initial vector is reset to all NULL bytes at card reset or whenever a non-SAM Command is called. The initial vector is changed after an CBC encryption/decryption command. When performing consecutive CBC encrypt/decrypt chaining, call this command once followed by consecutive CBC encrypt/decrypt commands. Call this command to reset the initial vector before a CBC encrypt/decrypt chain.

Specific Response Status Bytes:

SW1-SW2	Description
6986	No DF selected
6A86	Wrong P1, P1 must be 1 to 6
6700	Wrong P3, P3 must be 8 (or 0
6283	Current DF is blocked, or EF2 is blocked
6982	Security condition not satisfied
6A88	EF2 not found
6A83	Referenced Master Key in EF2 not found
6981	Invalid EF2 (FDB, MRL, etc not consistent)
6A87	Referenced KEY not capable of authentication
6983	Referenced Key is locked
9000	Target key generated, and ready in SAM memory

7.3. Encrypt

This command will encrypt data using DES or 3DES with either:

1. The session key created by the mutual authentication procedure with an ACOS2/3/6 card
2. A diversified key (secret code).
3. A bulk encryption key.
4. Encrypt the diversified secret code with the session key.
5. Prepare ACOS3 secure messaging command given a non-SM command.

CLA	80					
INS	74					
P1	**b7-b4**	**b3**	**b2**	**b1**	**b0**	**Description**
	-	0	0	0	-	ECB Mode
	-	0	0	1	-	CBC Mode
	-	0	1	0	-	Retail MAC Mode
	-	0	1	1	-	MAC Mode
	-	1	0	0	-	Prepare ACOS3 SM cmd.
	-	-	-	-	0	Triple DES
	-	-	-	-	1	Single DES
	0000	-	-	-	-	All other values – RFU
P2	P2 is derived key in SAM set using Load Key function: 1 – Encrypt Data with Session Key *Ks* 2 – Encrypt Data with Diversified Key *Sc* 3 – Encrypt Data with Bulk Encryption Key 0 – return ENC (*Sc*, *Ks*) If P1.b3 = 1, P2 must be 1					
P3	If P2 = 1-3, multiple of 8 up to 128 bytes If P2 = 0, 0					
Data	Plain text					

If the current encrypt function is to perform Prepare ACOS3 SM command, a non-SM command is expected in the command data. A Get Response Command will yield the complete secure messaging command. The entire response data can be sent to the ACOS3 card. Please see scenario in Section 9.8 for more information.

On success, the command will return SW = 61XX where XX indicate the length of the output is multiple of 8. Issue GET RESPONSE with P3 = XX to retrieve the result.

This command will compute a DES/3DES encryption using the derived key specified in P2, and the plain text command data. If DES mode is Single DES, the 2nd half of the derived key will be ignored. The key diversification must first be performed with Diversify Key (See Section 7.2). If encrypting data with Ks, mutual authentication must also be done.

Specific Response Status Bytes:

SW1-SW2	Description
6986	No DF selected
6A86	Invalid P1 or P2
6700	Incorrect P3
6A83	ACOS Target Key is not ready (use Diversify to generate the key)
61XX	Encryption is done, use GET RESPONSE to get the result

7.4. Decrypt

This command will decrypt data using DES or 3DES with either:

1. The session key created by the mutual authentication procedure with an ACOS2/3/6 card
2. A diversified key (secret code).
3. A bulk encryption key.
4. Decrypt the diversified secret code with the session key.
5. Verify and Decrypt ACOS3 secure messaging response.

CLA	80					
INS	76					
P1	**b7-b4**	**b3**	**b2**	**b1**	**b0**	**Description**
	-	0	0	0	-	ECB Mode
	-	0	0	1	-	CBC Mode
	-	1	0	0	-	Verify and Decrypt ACOS3 SM Response
	-	-	-	-	0	Triple DES
	-	-	-	-	1	Single DES
	0000	-	0	-	-	All other values - RFU
P2	P2 is derived key in SAM set using Load Key function: 1 – Decrypt Data with Session Key *Ks* 2 – Decrypt Data with Diversified Key *Sc* 3 – Decrypt Data with Bulk Encryption Key 0 – return DEC (*Sc*, *Ks*)					
P3	If P2 = 1-3, multiple of 8 up to 128 bytes If P2 = 0, 0					
Data	Ciphertext					

If the current decrypt function is to perform Verify and Decrypt ACOS3 SM response, a SM response is expected in the command data. The non-SM response will output after get response. Please see scenario in Section 9.8 for more information.

On success, the command will return SW = 61XX where XX indicate the length of the output is multiple of 8. Issue GET RESPONSE with P3 = XX to retrieve the result.

This command will compute a DES/3DES decryption using the derived key specified in P2, and the ciphertext command data. If DES mode is Single DES, the 2nd half of the derived key will be ignored. The key loading must first be performed with Load Key (See Section 7.2). If decrypting data with Ks, mutual authentication must also be done.

Specific Response Status Bytes:

SW1-SW2	Description
6986	No DF selected
6A86	Invalid P1 or P2
6700	Incorrect P3
6A83	ACOS Target Key is not ready (use Diversify to generate the key)
61XX	Decryption is done, use GET RESPONSE to get the result

7.5. Prepare Authentication

This command would authenticate the SAM card (as the terminal) to the ACOS2/3/6 or MIFARE UL-C card.

CLA	80
INS	78

P1	DES mode:
	LSb=0: Triple DES
	LSb=1: Single DES

P2	00 – ACOS2/3/6 authenticate 01 – MIFARE UL-C authenticate by (diversified) terminal key 05 – MIFARE UL-C Authenticate by Bulk Encryption Key
P3	08
Data	ACOS: Card Challenge Data RND_C UL-C: Card Challenge ek(RndB)

If P2 = 00_H: On success, the command will return SW1-SW2 = 6110_H. Issue GET RESPONSE with P3 = 10h to get:

ENC (RNDc, Kt) || RNDt

; where ENC is Triple DES or Single DES; and RND_T is generated by the SAM.

For more information, please see Section 9.2.

If P2 = 01_H or 05_H: On success, the command will return SW1-SW2 = 6110_H. Issue GET RESPONSE with P3 = 10_H to get:

ek(RndA + RndB')

Where RndA is the terminal challenge and RndB' is the decrypted card challenge obtained by rotating the original RndB left by 8 bits internally.

Note that UL-C authentication always uses triple DES. For more information, please see Sections 9.3 and 9.4.

Specific Response Status Bytes:

SW1-SW2	Description
6986	No DF selected
6A86	Invalid P1 or P2
6700	Incorrect P3, must be 0x08
6A83	ACOS Key (KT or KC) is not ready (use Diversify to generate this key)
6982	Security condition not satisfied
6110	Command completed, issue GET REPONSE to get the result

7.6. Verify Authentication

This command would verify the ACOS2/3/6 or MIFARE UL-C card to the terminal. The Session Key Ks would also be generated internally.

CLA	80
INS	7A

P1	DES mode:
	LSb=0: Triple DES
	LSb=1: Single DES

P2	00 – Verify ACOS2/3/6 authentication return value 01 – Verify MIFARE UL-C authentication return value.
P3	08
Data	ACOS: DES (K_S, RND_T) UL-C: ek(RndA’)

On success, the command will return SW1-SW2 = 9000_H. This means that the Mutual Authentication with client ACOS card is successful, and the client card is authenticated.

Specific Response Status Bytes:

SW1-SW2	Description
6986	No DF selected
6A86	Invalid P1 or P2
6700	Incorrect P3, must be 0x08
6A83	ACOS-SAM Session Key or RND_T are not ready. Use “Prepare Authentication” to build these keys.
6982	Data is incorrect
9000	Data is correct, ACOS Mutual Authentication is successful

7.7. Verify ACOS Inquire Account

This command would verify the ACOS2/3/6 card's Inquire Account purse command. It would verify that the MAC checksum returned by ACOS2/3/6 are correct with the SAM's diversified key. Please refer to Section 9.7.

CLA	80
INS	7C

P1	DES mode:
	Bit0=0: Triple DES
	Bit0=1: Single DES
	Bit1=1: ACOS INQ_AUT is enabled
	Bit2=1: ACOS INQ_ACC_MAC is enabled

P2	0
P3	0x1D
Data	Data Block returned by INQUIRE ACCOUNT of client ACOS card, see below

The Data Block to be sent to ACOS6-SAM has the following format:

	Reference Data	MAC	Transaction Type	Balance	ATREF	MAX Balance	$TTREF_C$	$TTREF_D$
Bytes:	4	4	1	3	6	3	4	4

The 1st 4 bytes is the challenge data sent to INQUIRE ACCOUNT. While the succeeding bytes are returned by INQUIRE ACCOUNT. The command will then verify if MAC[4] is correct, based on the other data fields and the current ACCOUNT KEY K_{ACCT}.

On success, the command will return SW1-SW2 = 9000, meaning the client card's PURSE information is authentic and un-tampered.

Specific Response Status Bytes:

SW1-SW2	Description
6986	No DF selected
6A86	Invalid P1 or P2
6700	Incorrect P3
6A83	ACOS Key K_S or K_{ACCT} are not ready; use DIVERSIFY command to generate K_{ACCT}; if applicable, use "Prepare Authentication" to generate K_S.
6F00	Data Block's MAC is incorrect
9000	Data Block's MAC is correct

7.8. Prepare ACOS Account Transaction

To create an ACOS2/3/6 Credit/Debit command, the MAC must be computed for ACOS2/3/6 to verify. Please refer to Section 9.10 and 9.11 for Debit and Credit respectively.

CLA	80
INS	7E

P1	DES mode:
	LSb=0: Triple DES
	LSb=1: Single DES
	Bit1=1: ACOS TRNS_AUT is enabled.

P2	E2: Credit E6: Debit
P3	0x0D
Data	Data Block

The Data Block to be sent to the SAM has the following format:

	Amount	$TTREF_C$ / $TTREF_D$	ATREF
Bytes:	3	4	6

Amount	- amount to CREDIT / DEBIT
TTREF	- if P2 = E2, use $TTREF_C$; if P2 = E6, use $TTREF_D$
ATREF	- this is the ATREF field returned by the INQUIRE ACCOUNT command

On success, SW1-SW2 = 610B, the Data block returned by SAM has the following format:

	MAC	Amount	TTREF
Bytes:	4	3	4

The return data block can be transferred to the ACOS3/6 client card in DEBIT/CREDIT commands.

Specific Response Status Bytes:

SW1-SW2	Description
6986	No DF selected
6A86	Invalid P1 or P2
6700	Incorrect P3, must be 0x0D
6A83	ACOS Key K_S or K_{ACCT} are not ready; use DIVERSIFY command to generate K_{ACCT}; if applicable, use “Prepare Authentication” to generate K_S.
610B	Command completed, issue GET REPONSE to get the result

7.9. Verify Debit Certificate

For ACOS3/6, if the DEBIT command has P1 = 1, a debit certificate is returned. The debit certificate can be checked by comparing the ACOS3 response to the result of this command.

CLA	80
INS	70

P1	DES mode:
	Bit0=0: Triple DES
	Bit0=1: Single DES
	Bit1=1: TRNS_AUT is enabled, the cipher with Session Key

P2	0
P3	0x14
Data	Data Block

Data Block format is:

	MAC	Amount	New Balance	ATREF	$TTREF_D$
Bytes:	4	3	3	6	4

MAC	Debit Certificate returned by ACOS3 DEBIT command
AMOUNT	Amount last debited from card
NEW BALANCE	expected new balance after the DEBIT
ATREF	ATREF used before the last DEBIT command
$TTREF_D$	$TTREF_D$ used in the last DEBIT command

Specific Response Status Bytes:

SW1-SW2	Description
6986	No DF selected
6A86	Invalid P1 or P2
6700	Incorrect P3, must be 0x14
6A83	ACOS Key K_S or K_{ACCT} are not ready; use DIVERSIFY command to generate K_{ACCT}; if applicable, use "Prepare Authentication" to generate K_S.
6982	Security condition not satisfied
6F00	DEBIT CERTIFICATE is invalid
9000	Success, DEBIT CERTIFICATE is valid

7.10. Get Key

Get key allows secure key injection from the current SAM's Key File (SFI=02) into another ACOS6/ACOS6-SAM with or without key diversification. Using this ensures the keys to be injected is protected by encryption and message authentication codes.

If bit 7 of the Special Function Flag (Key Injection Only Flag) of the Card Header Block (Section 3.2) has been set and the key file has been activated, Get Key must be used for loading or changing keys in the card. Setting this bit will disable Read Record command for the key file under any circumstances after activation.

Before this command is to be executed, a session key is already established with the target card with the mutual authentication procedure of Section 5.3.

Remark: Get Key command can only get the Key data

CLA	80
INS	CA
P1	00 – Response data is Key in MSAM 01 – Response data is Diversify Key
P2	Key ID
P3	If P1=00, P3 is 08 If P1=01, P3 is 10
Data	If P1=00, command data is RND_{Target} If P1=01, command data is RND_{Target} + serial (or batch) number of target sam

The security condition of Get Key is the Read/Get Key access condition of KEY File.

A successful execution of this command will yield 0x611C. A Get Response would yield the encrypted key as:

RND_{Source} + Encrypted Key Data + MAC

The generation of Encrypted Key Data and MAC follows Section 5.9. For more information, please see the scenario in Section 9.12.

Specific Response Status Bytes:

SW1-SW2	Description
6985	SAM Session Key not ready
6283	Current DF is blocked, or Target EF is blocked
6986	No DF selected
6981	Wrong file type of Key file, it should be Internal Linear Variable File
6982	Target file's header block has wrong checksum, or security condition not satisfied
6A86	Invalid P1 or P2
6700	Incorrect P3
6A83	Target Key is not ready or Key Length less than 16
611C	Success, use GET RESPONSE to get the result

8.0. Response Status Bytes

This section lists all the card response status bytes SW1-SW2 used in ACOS6-SAM and shows their general meaning. This section is meant for quick references. For meanings specific to a command, please see the corresponding command section.

SW1-SW2	Meaning
90 00	Command executed successfully
91XX	User file selected successfully in ACOS2 mode
61XX	XX encodes the number of data bytes available. Issue GET RESPONSE with P3=XX to retrieve data
6282	Invalid offset
6283	Selected file terminated / deactivated
63CX	Wrong authentication / verification data, X tries left for authentication key / PIN.
6400	Target file is terminated
6700	Wrong P3, P3 not compatible with P1/P2
6966	PIN_ALT of PIN is disabled
6981	Command incompatible with file structure
6982	Security status not satisfied or file header checksum incorrect
6983	Referenced PIN is locked
6985	Conditions of use not satisfied / no data available / MAC computation error in secure messaging (random number or session key not ready)
6986	Command not allowed (no currently selected DF/EF) / no MF on card
6988	Wrong MAC is submitted in secure messaging
6A80	Incorrect parameters in the command data field
6A82	File or application not found / card is in user terminated state
6A83	Record / Key not found
6A84	Not enough memory or record space
6A86	Incorrect parameters P1/P2
6A87	Key referenced cannot perform required authentication.
6A88	Reference data / file not found
6A89	File already exists
6B00	Wrong parameters P1/P2
6CXX	Wrong P3, XX encodes the exact number of available data bytes
6D00	Invalid INS
6E00	Invalid CLA
6F00	Invalid physical address, EEPROM error, command not available.

Table 25: Response Status Bytes

9.0. SAM Scenarios

This section outlines the typical scenarios of using an ACOS6 SAM cards along with an ACOS2/3/6 client card. The ACOS6 SAM should already be loaded with a file system and keys similar to that of Section 10.0 or 11.0. This section will demonstrate in APDU form how to perform several SAM and client card interaction. Notice that many scenarios that are demonstrated with ACOS2 / 3 client card also work with ACOS6 client card with some modifications to the command set.

9.1. Personalizing ACOS2/ACOS3 KEYS / Codes

You can use the ACOS6-SAM to set the diversified keys or secret codes of ACOS2/3. If the SAM has N Master keys, you can use them to generate your keys. In the example below, SAM Master Key$_I$ is used to generate ACOS Key$_J$. If 3DES key is needed, please follow Section 7.1 to generate the first and second half of the 16-byte 3DES key by multiple issuance of this command.

For ACOS6 client card, similar procedure can be followed. Alternatively, the secure key injection procedure of Section 9.12 can be followed.

SAM	Terminal	ACOS2 / 3
	< Select Issuer DF < 00 A4 00 00 02 < XX XX < Submit Issuer PIN < 00 20 00 01 08 < XX XX…XX	
	Get Card Serial Number > (ACOS3) 80 14 00 00 08 > (ACOS2) 80 A4 00 00 02 FF 00 > (ACOS2) 80 B2 00 00 08 >	< Serial Number < 9000 < Serial Number
Check if Issuer PIN is submitted; Check if referenced master key is valid; Compute Key 6108 >	< Generate KEYi < 80 88 00 K$_{MI}$ 08 < Serial Number	
Generated Key >	< Get Response < 00 C0 00 00 08	
	Submit IC Code: 80 20 07 00 08>< 8-byte IC Code > Select FF 03 80 A4 00 00 02 FF 03 > Set KEYj > Write key record with KEYj > 80 D2 <KEYj record> 00 08 > Generated Key >	< 9000 < 9000 < 9000

Figure 15: Personalizing ACOS2/3 keys

9.2. Mutual Authentication with ACOS

The terminal will now use the SAM to compute the ACOS2/3/6 Session Key, and perform Mutual Authentication with the ACOS2/3 card. The SAM should know the Terminal Key and Card Key of the given ACOS2/3 since it is assumed that the keys are generated by the SAM.

SAM	Terminal	ACOS2 / 3 / 6
	Get Card Serial Number > (ACOS3) 80 14 00 00 08 > (ACOS2) 80 A4 00 00 02 FF 00 > (ACOS2) 80 B2 00 00 08 >	< Serial Number < 9000 < Serial Number
Generate Card Key using Serial Number 9000 >	< Generate Card Key using Master Key$_I$ < Diversify (@K_{MI}, Serial Number) < 80 72 04 # K_{MI}, 08 < Serial Number	
Generate Terminal Key using Serial Number 9000 >	< Generate Terminal Key using Master Key$_J$ < Diversify (@K_{MJ}, Serial Number) < 80 72 03 #K_{MJ} 08 < Serial Number	
	Get Challenge > 80 84 00 00 08 >	< RNDc
1. Generate RNDt 2. Compute R = 3DES (RNDc, Kt) 3. Compute Session Key Ks KsL = 3DES (3DES (RNDc, Kc), Kt) KsR = 3DES (RNDt, REV (Kt)) 4. Form Response Block = R || RNDt 6110 >	< Prepare Authentication < 80 78 00 00 08 < RNDc	
R || RNDt >	< Get Response < 00 C0 00 00 10	
	Authenticate (R, RNDt) > 80 82 00 00 10 > R > RNDt >	1. Compute Ks KsL = 3DES (3DES (RNDc, Kc), Kt) KsR = 3DES (RNDt, REV (Kt)) 2. R1 = 3DES (RNDt, Ks) 3. Get Response Block = R1 < 6108
	Get Response > 80 C0 00 00 08 >	< R1
	< Verify ACOS Authentication	

Check if R1 == DES (RNDt, Ks) If OK, 9000 >	< 80 7A 00 00 08 < R1 Proceed if both Card and Terminal are Authenticated	

Figure 16: Mutual Authentication with ACOS2/3 cards

9.3. 3 Pass Authentication with Ultralight C with Static Key

The following is the authentication flow with a MIFARE Ultralight-C card using a static key.

SAM	Terminal	Mifare ULC
9000 >	< Generate bulk encryption Key using Master Key$_J$ < 80 72 05 #K$_{MJ}$ 00	
	Start Authenticate > 1A 00 >	< ek(RndB)
6110 >	< Prepare Authentication < 80 78 00 05 08 < ek(RndB)	
ek(RndA + RndB') >	< Get Response < 00 C0 00 00 10	
	Authenticate (R, RNDt) > AF ek(RndA + RndB') >	< 00 ek(RndA') 9000
Check RndA If OK, 9000 > If fail, 6982 >	< Verify Authentication < 80 7A 00 01 08 < ek(RndA')	

Figure 17: Mutual Authentication with MIFARE Ultralight C, Key not Diversified

9.4. 3 Pass Authentication with Ultralight C with Diversified Key

The following is the authentication flow with a MIFARE Ultralight-C card using a diversified key.

SAM	Terminal	Mifare ULC
	Retrieve UL-C's UID Use as diversification data	<- UID
9000 >	< Generate Terminal Key using Master Key$_J$ < 80 72 03 #K$_{MJ}$ 08 < 8 byte diversification data	
	Start Authenticate > 1A 00 >	< ek(RndB)
6110 >	< Prepare Authentication < 80 78 00 01 08 < ek(RndB)	
ek(RndA + RndB')>	< Get Response < 00 C0 00 00 10	
	Authenticate (R, RNDt) > AF ek(RndA + RndB')>	< 00 ek(RndA') (9000)
Check RndA If OK, 9000 > If fail, 6982 >	< Verify Authentication < 80 7A 00 01 08 < ek(RndA')	

Figure 18: Mutual Authentication with MIFARE Ultralight C, Key Diversified

9.5. Submit PIN (ciphered)

Assuming Mutual Authentication is already performed, and Session Key Ks is ready in SAM.

SAM	Terminal	ACOS2 / 3
	Get Card Serial Number >	< Serial Number
Compute SC using Kmi and Serial Number	< Use SAM Master keyi to compute Secret Code < Diversify (@Kmi, Serial Number) < 80 72 01 #kmi 08 < Serial Number	
Compute R = ENC(SC, K_S) 6108 >	< Encrypt command < 80 74 00 00 00 00	
R >	< Get Response < 00 C0 00 00 08	
	Submit encrypted PIN > 80 20 06 00 08 > R >	< 9000

Figure 19: Submit Enciphered PIN for ACOS2/3

9.6. Change PIN (ciphered)

- assuming Mutual Authentication is already performed, and Session Key Ks is ready in SAM.

SAM	Terminal	ACOS2 / 3
	Ask for Current PIN	
	Submit PIN > 80 20 06 00 08 > PIN >	< 9000
	Ask for new PIN	
Compute R = DEC (new PIN, Ks) Get Response Block = R 6108 >	< Decrypt (@Ks, new PIN) < 80 76 00 01 08 < newPIN	
R >	< Get Response < 00 C0 00 00 08	
	Change PIN (R) > 80 24 00 00 08 > R >	< 9000

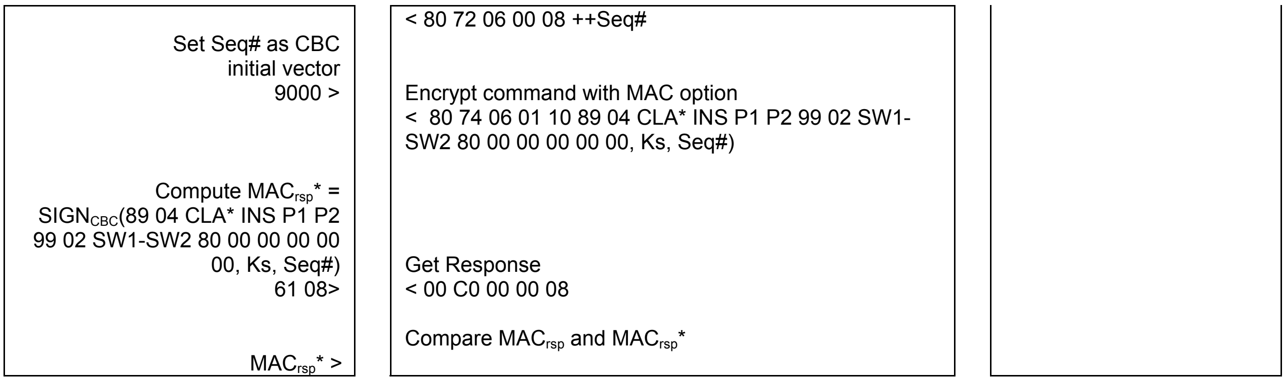
Figure 20: Change Enciphered PIN for ACOS2/3

9.7. Secure Messaging Computation for ACOS3

Secure Messaging is supported by ACOS3 version 1.07 or above. Assuming Mutual Authentication is already performed, a 3DES Session Key Ks is ready in SAM, and RND_C memorized in the SAM. This section demonstrates how the terminal can initiate an ISO-IN command to the ACOS3 card with secure messaging and using ACOS6 SAM as the secured encryption engine.

See ACOS3 reference manual for more information about secure messaging.

SAM	Terminal	ACOS 3 v1.07 or above
	Terminal wish to send to ACOS3: 80 INS P1 P2 P3 <Cmd Data>	
Set Seq# as CBC initial vector 9000 > Compute: <Enc Data> = ENC_{CBC}(<Cmd Data> Padding, Ks, Seq#) 61 XX> <Enc Data> >	**Encrypt command data** Set Seq# = 00 00 00 00 00 00 FF FF AND RND_C + 1 Load Key Data to Load CBC Initial Vector. < 80 72 06 00 08 Seq# Encrypt command with CBC option: < 80 74 02 01 XX <Cmd Data> Padding Get Response < 00 C0 00 00 XX	Set Seq# = 00 00 00 00 00 00 FF FF AND RND_C + 1
Set Seq# as CBC initial vector 9000 > MAC_{cmd} = $SIGN_{CBC}$(89 04 8C INS P1 P2 87 L_{87} Pi <Enc Data> Padding2, Ks, Seq#) 61 08> MAC >	**Compute MAC for secure messaging** < Load Key Data to Load CBC Initial Vector. < 80 72 06 00 08 Seq# Set Pi = Number of padding bytes used in Padding P3* = P3 + Pi + 9 L_{87} = P3 + Pi + 1 Encrypt command with MAC option < 80 74 06 01 XX 89 04 8C INS P1 P2 87 L_{87} Pi <Enc Data> Padding2 Get Response < 00 C0 00 00 08	
	Send secure messaging command 8C INS P1 P2 P3* 87 L_{87} Pi <Enc Data> 8E 04 MAC_{Cmd}> Get Response 80 C0 00 00 0A >	Verify MAC_{cmd} Decrypt <Enc Data> Perform INS. Set Seq# ++ Compute MAC_{rsp} = $SIGN_{CBC}$(89 04 CLA* INS P1 P2 99 02 SW1-SW2 80 00 00 00 00 00, Ks, Seq#) < 61 0A < 99 02 SW1-SW2 8E 04 MAC_{rsp}
	Verify MAC_{rsp} Load Key Data to Load CBC Initial Vector.	



Set Seq# as CBC initial vector
9000 >

Compute $MAC_{rsp}*$ = $SIGN_{CBC}$(89 04 CLA* INS P1 P2 99 02 SW1-SW2 80 00 00 00 00 00, Ks, Seq#)
61 08>

$MAC_{rsp}*$ >

< 80 72 06 00 08 ++Seq#

Encrypt command with MAC option
< 80 74 06 01 10 89 04 CLA* INS P1 P2 99 02 SW1-SW2 80 00 00 00 00 00, Ks, Seq#)

Get Response
< 00 C0 00 00 08

Compare MAC_{rsp} and $MAC_{rsp}*$

ENC_{CBC}(data, key, initial vector) and $SIGN_{CBC}$(data, key, initial vector) are CBC Encryption/MAC with data, key and initial vector as input.

Figure 21: Secure Messaging Computation for ACOS3 – Method 1

9.8. Secure Messaging Computation for ACOS3 – Method 2

In ACOS6-SAM revision 4.01 or higher, a more efficient method of computing ACOS3 secure messaging is available. This method lessen the work load of the terminal as well. As in the Section 9.7, mutual authenticate with the client ACOS3 card is already performed.

ACOS6-SAM v4.01 or above	Terminal	ACOS3 v1.07 or above
	Terminal wish to send [ACOS3 command] to ACOS3: = 80 INS P1 P2 P3 <Cmd Data> [ACOS3 command] must not be > 128 bytes	
	Encrypt command data	
	< Encrypt, ACOS3 SM < 80 74 08 01 Le [ACOS3 command]	
61*xx* >		
	< Get Response < 00 C0 00 00 *xx*	
[SM ACOS3 Command] >		
	[SM ACOS3 Command] >	
		< 61*xx*
	Get Response > 00 C0 00 00 *xx* >	
		< *[Encrypt Response and MAC]*
	< Decrypt, ACOS3 SM < 80 76 08 01 Le *[Encrypt Response and MAC]*	
61*xx* >		
	< Get Response < 00 C0 00 00 *xx*	
[Response Data] 9000 >		

Figure 22: Secure Messaging Computation for ACOS3 – Method 2

9.9. Inquire Account

SAM	Terminal	ACOS 2 / 3
	Get Card Serial Number >	< Serial Number
Generate Certify Key Kacct KacctL = 3DES (SN, Km) KacctR = 3DES (SN, Km) 9000 >	< Diversify (@Km, Serial Number) < 80 72 02 #Km 08 < Serial Number	
	Inquire Account (@Kcrt) > 80 E4 02 00 04 > Ref >	< 6119
	Get Response > 80 C0 00 00 19 >	< MAC, TransType, Bal, ATREF, Max, TTREFc, TTREFd,
Compute MAC from Kacct and Datain, then compare with MAC in Datain R = 3MAC (Data, Kacct) If R == MAC 9000 >	< Verify Inquire Account < 80 7C 00 00 1D < Ref, MAC, TransType, Balance, ATREF, Max, TTREFc, TTREFd [4]	

Figure 23: Inquire Account of ACOS2/3

9.10. Debit

SAM	Terminal	ACOS 2 / 3
	Get Card Serial Number >	< Serial Number
Generate Debit Key Kaact KacctL = 3DES (SN, Km) KacctR = 3DES (SN, Km) 9000 >	< Diversify (@Km, Serial Number)	
	Inquire Account (@Kd) > 80 E4 00 00 04 > Ref >	< 6119
	Get Response > 80 C0 00 00 19 >	< MAC, TransType, Bal, ATREF, Max, TTREFc, TTREFd,
++ATREF Data = E6, Amount, TTREFd, ATREF, 0, 0 R = 3MAC (Kacct, Data) R1 = R || Amount || TTREFd Response Block = R1 610B >	< Prepare ACOS Transaction < 80 7E 00 E6 0x0D < Amount, TTREFd, ATREF	
R1 >	< Get Response < 00 C0 00 00 0B	
	Debit > 80 E6 01 00 0B > R1 >	< Perform Debit and return Debit Certificate < 6104
	Get Reponse> 80 C0 00 00 04>	< Debit Certificate
Check if DC is correct 9000 >	< Verify Debit Certificate < 00 70 00 00 04 < DC, AMT, New BAL, ATREF, TTREFd	

Figure 24: Debit ACOS2/3 Purse

9.11. Credit

SAM	Terminal	ACOS 2 / 3
	Get Card Serial Number >	< Serial Number

Generate Credit Key Kaact
 	
KacctL = 3DES (SN, Km)
KacctR = 3DES (SN, Km,)
 	
9000 >
	< Diversify (@Km, Serial Number)	

	Inquire Account (@Kd) > 80 E4 01 00 04 > Ref >	< 6119
	Get Response > 80 C0 00 00 19 >	< MAC, TransType, Bal, ATREF, Max, TTREFc, TTREFd,

++ATREF
 	
Data = E2, Amount, TTREFc, ATREF, 0, 0
 	
R = MAC (Data, Kacct)
 	
R1 = R || Amount || TTREFc
 	
Get Response Block = R1
 	
610B >
	< Prepare ACOS Transaction
	< 00 7E 00 E2 0D < Amount, TTREFd, ATREF	

R1 >	< Get Response < 00 C0 00 00 0B	
	Credit > 80 E2 00 00 0B > R1 >	< 9000
	Perform Verify Inquire Account with Credit Key and new amount Inquire Account (@Kc) > 80 E4 01 00 04 > Ref >	< 6119
	Get Response > 80 C0 00 00 19 >	< MAC, TransType, Bal, ATREF, Max, TTREFc, TTREFd,
Compute MAC from Kacct and Datain, then compare with MAC in Datain Data = Ref, Trans Type, Bal, ATREF, 00, 00, TTREFc, TTREFd	< Verify Inquire Account < 00 7C 00 00 1D < Ref, MAC, TransType, Balance, ATREF, Max, TTREFc, TTREFd [4]	

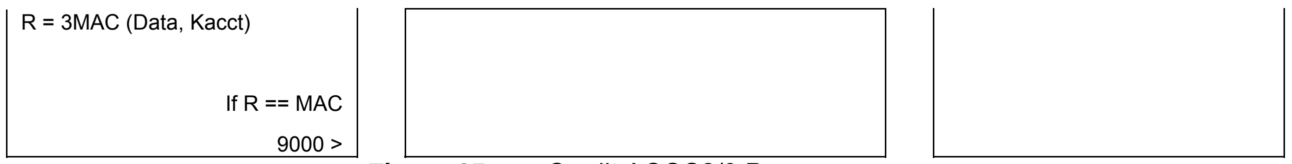


Figure 25: Credit ACOS2/3 Purse

9.12. Key Injection

Key injection can be used to securely load a key or diversified key into a target ACOS6-SAM or client ACOS6 card. In the following scenario, we will assume both SAM cards already shared a set of mutual authentication keys and mutual authentication (of Section 9.2) has already been performed successfully.

MSAM (ACOS6 SAM)	Terminal	SAM (ACOS6)
Personalize the MSAM Card		Personalize the SAM Card
	< Get Access Right of Get Key	
Return OK >		
	Read Key Version Number from SAM >	
		< Return *Key Version Number*
	< Generate Card Key using K_{MI} < 80 72 04 # K_{MI} 08 < *Key version Number*	
Generate Card Key 9000 >		
	< Generate Terminal Key using K_{MJ} < 80 72 03 # K_{MJ} 08 < *Key version Number*	
Generate Terminal Key 9000 >		
	GET CHALLENGE > 00 84 00 00 08 >	
		< Generate and return *RNDc*
	< Pre ACOS Authenticate with *RNDc* < 80 78 00 00 08 *RNDc*	
Compute Session Key Ks *R = 3DES (RNDc, Kt) || RNDt*		
Return *R* >		
	Mutual Authenticate > 00 82 Kc Kt 10 *R*	
		Compute Session Key Ks *R1 = 3DES (RNDt, Ks)* < Return *R1*
	< Verify ACOS Authentication < 80 7A 00 00 08 *R1*	
Check *R1* Return OK >		
	GET CHALLENGE > 00 84 00 00 08 >	
	< Get Key from MSAM with Diversify < 80 CA 01 *Ki* 10 *RNDsam* || *Key version Number*	< Generate and return *RNDsam*
Compute *ENCkey* Compute *MAC* Generate *RNDmsam* *RNDmsam* | *ENCkey* | *MAC* >		
	Set Key > 80 DA 00 Kj 1C *RNDmsam* | *ENCkey* | *MAC* >	
	Repeat this procedure as necessary to inject other keys.	< 9000

Figure 26: Key Injection to ACOS6 / ACOS6-SAM

10.0. Quick Start Guide

This quick start guide is to help the application developer to get familiar with the ACOS6-SAM. In order to use the SAM card, first it has to be initialized. The initialization entails creating a file system on the card and loading a master key set. The application developer will first have to be familiarized with the ACOS6-SAM commands and file system. Then to design the file system and key set based on the application needs. To help the application developer shorten this process, this document aims to guide the user through the initialization and go through some of the authentication procedures.

10.1. ACOS3 Client Card Sample

The included script works with ACS Script Tools and it initializes the card with a standard file structure and key set. Depending on the application needs, this script can be modified to suit the users' needs. The script will generate the following file system.

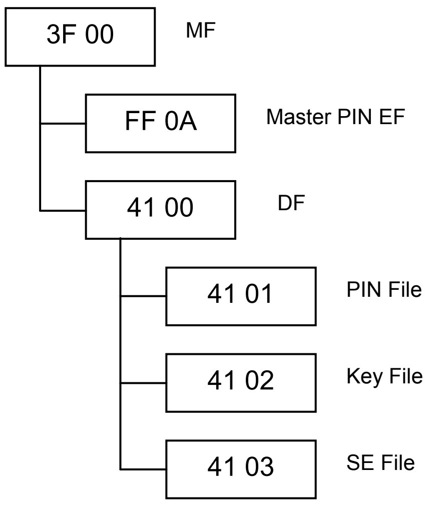


Figure 27: SAM sample script file system

The script listed below will generate the file system on a blank card. It is important to note that this script can be ran only once. Once the MF is created, it can not be removed unless the CLEAR CARD command is called and the card life cycle fuse (in Section 3.1) has not been set.

```
; create MF
00 E0 00 00 0E 62 0C 80 02 2C 00 82 02 3F FF 83 02 3F 00 (9000)

; Create EF1 to store the PIN
; FDB=0C, MRL=0A, NOR=3, READ=NONE, WRITE=IC
; READ=NEVER, WRITE=IC
00 E0 00 00 1B 62 19 83 02 FF 0A 88 01 01 82 06 0C 00 00 0A 00 03 8C 08 7F FF FF FF
FF 27 27 FF (9000)

; GLOBAL PINS
; change global PIN1 to diversify
00 DC 01 04 0A 01 88 12 12 12 12 12 12 12 12 (9000)
00 DC 02 04 0A 02 88 22 22 22 22 22 22 22 22 (9000)
00 DC 03 04 0A 03 88 33 33 33 33 33 33 33 33 (9000)

;***************************************************************************
```

```
; Create next DF DRT01: 4100
00 E0 00 00 2B 62 29 82 01 38 83 02 41 00 8A 01 01 8C 08 7F 03 03 03 03 03 03 03 8D
02 41 03 80 02 03 20 AB 0B 84 01 88 A4 06 83 01 01 95 01 08 (9000)

; create PIN FILE EF1 4101
00 E0 00 00 1C 62 1A 82 05 0C 01 00 12 01 83 02 41 01 88 01 01 8A 01 01 8C 07 6F 03
03 03 03 03 03 (9000)

; APPEND RECORD TO EF1, define 1 PIN record in EF1
00 E2 00 00 0A 81 88 11 22 33 44 55 66 77 88 (9000)

; Create KEY FILE EF2 4102
00 E0 00 00 1D 62 1B 82 05 0C 41 00 16 03 83 02 41 02 88 01 02 8A 01 01 8C 08 7F 03
03 03 03 03 03 03 (9000)

; APPEND RECORD TO EF2, define 3 KEY records in EF2 - MASTER KEYS
; Maximum of 3 MASTERKEYS for this file

;1st Master key, key ID=81, key type=03 int/ext authenticate, usage counter=FF FF,
retries=8:retries left=8,
;algo reference=00 (3DES), master key = change 11 .... 11 (16-bytes for 3DES)with
your own key
00 E2 00 00 16 81 03 FF FF 88 00 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11
(9000)

;2nd Master key, key ID=82, key type=02 internal authenticate, usage counter=FF FF,
;algo reference=00 (3DES), master key = change 11 .... 11 (16-bytes for 3DES)with
your own key
00 E2 00 00 15 82 02 FF FF 00 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 (9000)

;3rd Master key, key ID=83, key type=01 external authenticate, retries=8:retries
left=8,
;algo reference=01 (DES), master key = change 11 .... 11 (8 bytes for DES)with your
own key
00 E2 00 00 0C 83 01 88 01 11 11 11 11 11 11 11 11 (9000)

; Create SE FILE 4103 - SE
00 E0 00 00 1D 62 1B 82 05 0C 01 00 27 05 83 02 41 03 88 01 03 8A 01 01 8C 08 7F 03
03 03 03 03 03 03 (9000)

; APPEND RECORD to SE
; SE#1
00 E2 00 00 0B 80 01 01 A4 06 83 01 81 95 01 80 (9000)
; SE#2
00 E2 00 00 0B 80 01 02 A4 06 83 01 81 95 01 08 (9000)
; SE#3
00 E2 00 00 0B 80 01 03 A4 06 83 01 07 95 01 08 (9000)

;*****************************************************************************

; Activate all files.

00 44 00 00 02 41 03 (9000)
00 44 00 00 02 41 02 (9000)
00 44 00 00 02 41 01 (9000)

; activate 4100 DF
00 a4 00 00 00 (61xx)
00 44 00 00 02 41 00 (9000)
```

10.1.1. Example of Diversified Key Generation

The following is an example of how to use such SAM card with the above personalization and keyset to generate diversified keys to an end user card. This example will generate 3DES keys for card key and terminal key on an new ACOS3 card.

```
<SAM 00 A4 00 00 02 41 00
>SAM 61 2D

<SAM 00 20 00 01 08 12 12 12 12 12 12 12 12
>SAM 90 00

<Card 80 14 00 00 08
>Card 02 57 43 16 03 11 59 3C 90 00

;Generate Card key left half using SAM key 81
<SAM 80 88 00 81 08 02 57 43 16 03 11 59 3C
>SAM 61 08

<SAM 00 C0 00 00 08
>SAM 46 46 42 89 A2 DA 35 DA 90 00

;Generate Card key right half
<SAM 80 88 00 81 08 FD A8 BC E9 FC EE A6 C3
>SAM 61 08

<SAM 00 C0 00 00 08
>SAM 31 0C 4F E3 4B 35 39 9D 90 00

;Generate terminal key left half using SAM key 82
<SAM 80 88 00 82 08 02 57 43 16 03 11 59 3C
>SAM 61 08

<SAM 00 C0 00 00 08
>SAM 46 46 42 89 A2 DA 35 DA 90 00 (Same as Kt left half because key 81 and 82 are
the same)

;Generate terminal key right half
<SAM 80 88 00 82 08 FD A8 BC E9 FC EE A6 C3
>SAM 61 08

<SAM 00 C0 00 00 08
>SAM 31 0C 4F E3 4B 35 39 9D 90 00

; Write to the ACOS3 card
<Card 80 20 07 00 08 41 43 4F 53 54 45 53 54
>Card 90 00

<Card 80 A4 00 00 02 FF 03
>Card 90 00

<Card 80 D2 02 00 08 46 46 42 89 A2 DA 35 DA        ; Kc left half
>Card 90 00

<Card 80 D2 03 00 08 46 46 42 89 A2 DA 35 DA        ; Kt left half
>Card 90 00

<Card 80 D2 0C 00 08 31 0C 4F E3 4B 35 39 9D        ; Kc right half
>Card 90 00

<Card 80 D2 0D 00 08 31 0C 4F E3 4B 35 39 9D        ; Kt right half
>Card 90 00
```

10.1.2. Example of Mutual Authentication with SAM

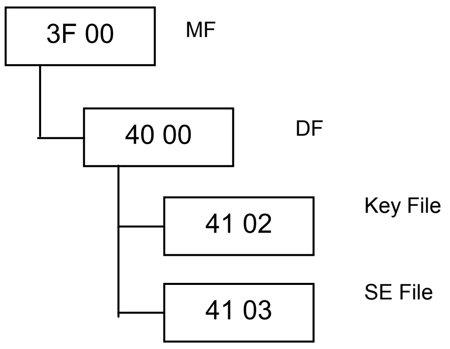
The following is a mutual authentication example showing how the ACOS6-SAM interacts with an ACOS3 card. Remember that the 3DES options register must be set in ACOS3 Personalization File FF 02 in order for this to work.

```
<SAM 00 A4 00 00 02 41 00
>SAM 61 2D
```

```
<SAM 00 20 00 01 08 12 12 12 12 12 12 12 12
>SAM 90 00

<CARD 80 14 00 00 08
>CARD 02 57 43 16 03 11 59 3C 90 00

<SAM 80 72 03 82 08 02 57 43 16 03 11 59 3C
>SAM 90 00

<SAM 80 72 04 81 08 02 57 43 16 03 11 59 3C
>SAM 90 00

<CARD 80 84 00 00 08
>CARD FA 1E 9B 9B 6E C5 1C F4 90 00

<SAM 80 78 00 00 08 FA 1E 9B 9B 6E C5 1C F4
>SAM 61 10

<SAM 00 C0 00 00 10
>SAM 52 C0 49 28 D4 02 CB 95 54 D1 A2 24 3C F0 28 D9 90 00

<CARD 80 82 00 00 10 52 C0 49 28 D4 02 CB 95 54 D1 A2 24 3C F0 28 D9
>CARD 61 08

<CARD 80 C0 00 00 08
>CARD 05 48 E3 8D 21 EB 6A E2 90 00

<SAM 80 7A 00 00 08 05 48 E3 8D 21 EB 6A E2
>SAM 90 00
```

10.2. Short Key External Authentication Sample

This sample initialization script demonstrate short key external authentication.



```
;Create File MF
00 E0 00 00 13 62 11 82 01 3F 83 02 3F 00 8A 01 01 8D 02 00 03 8C 01 00 (9000)

;Create File DF
00 E0 00 00 1B 62 19 82 01 38 83 02 40 00 8A 01 01 8D 02 40 03 8C 01 00 AB 06 88 01
80 9E 01 01 (9000)

;Create File Key File
00 E0 00 00 13 62 11 82 05 0C 00 00 18 05 83 02 40 02 8A 01 01 8C 01 00 (9000)
;Ext Key for special Auth
00 E2 00 00 14 81 08 55 00 11 22 33 44 55 66 77 88 99 AA BB CC DD EE FF 00 (9000)
;Int Key for diversify and Generate
00 E2 00 00 15 82 02 FF FF 00 00 11 22 33 44 55 66 77 88 99 AA BB CC DD EE FF (9000)
;Ext Key for normal Auth
00 E2 00 00 14 83 01 33 00 FF 00 11 22 33 44 55 66 77 88 99 AA BB CC DD EE (9000)
```

```
;Create File SE File
00 E0 00 00 13 62 11 82 05 0C 00 00 10 05 83 02 40 03 8A 01 01 8C 01 00 (9000)
;Append Record
00 E2 00 00 0B 80 01 01 A4 06 83 01 81 95 01 80 (9000)

;Select 4000
00 a4 00 00 02 40 00 (61xx)
;Activate 4000
00 44 00 00 00 (9000)
```

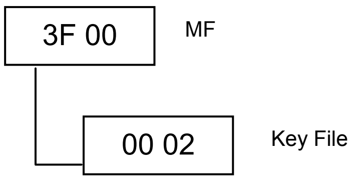
10.2.1. Example of Short Key External Authenticate

This demonstrate how short key authenticate is used with the above example.

```
;Get Challenge
<SAM 00 84 00 00 04
>SAM 94 5E 48 9C (9000)

;Ext Auth
;DATA_IN = 00 00 00 00 94 5e 48 9c
;KEY = 11 22 33 44 55 66 77 88 99 aa bb cc dd ee ff 00
;3DES (DATA_IN, KEY) = e8 a1 14 8B f1 03 3c a0
;DATA = Left(4 byte, 3DES (DATA_IN, KEY)    ) = e8 a1 14 8B
<SAM 00 82 00 81 04 e8 a1 14 8B
>SAM 9000
```

10.3. MIFARE Ultralight C Sample

The following initialization just creates one key file to demonstrate the MIFARE Ultralight C



```
;Create MF
00 E0 00 00 1A 62 18 82 01 3F 83 02 3F 00 8A 01 01 8C 08 7F FF FF FF FF FF FF FF 8D
02 00 03 (9000)

;Create Key File
00 E0 00 00 1D 62 1B 82 05 0C 00 00 16 05 83 02 00 02 88 01 02 8A 01 01 8C 08 7F FF
FF FF FF FF 01 FF (9000)

;Key 1, Ext Control Key
00 E2 00 00 14 81 01 33 00 8C A6 4D E9 C1 B1 23 A7 35 55 50 B2 15 0E 24 51 (9000)

;Key 2, Master Key for Ultra Light C
00 E2 00 00 15 82 06 FF FF 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
(9000)

;Key 3, Internal Control Key
00 E2 00 00 15 83 02 FF FF 00 89 B0 7B 35 A1 B3 F4 7E 09 E9 22 89 0D 6F 9C A1
(9000)

;Key 4, Init Key in Ultra Light C
00 E2 00 00 15 84 06 FF FF 00 49 45 4D 4B 41 45 52 42 21 4E 41 43 55 4F 59 46
(9000)
```

10.3.1. Example of Getting the Diversify Key

The following two commands gets the ACOS6 SAM diversified key from the SAM card

```
<SAM 80 88 01 82 08 <8-byte SN>
>SAM 6110
```

```
<SAM 00 C0 00 00 10
>SAM <16-byte Diversified Key> 9000
```

10.3.2. Example of APDU flow of UL-CAuthenticate with static key

The following authentication step follows the scenario based on Section 9.3. This procedure is useful in initializing MIFARE UL-C client cards with a diversified key.

```
; ACOS6-SAM Card - load the bulk encryption key local key 4 to RAM
<SAM 80 72 05 84 00
>SAM 9000

; UL-C Authenticate step 1
<UL-C 1A 00
>UL-C AF <ek(RndB)> 9000

; Prepare Authenticate using bulk encryption Key
<SAM 80 78 00 05 08 <ek(RndB)>
>SAM 6110

<SAM 00 C0 00 00 10
>SAM <ek(RndA + RndB')> 9000

; UL-C Authenticate command Step 2
<UL-C AF <ek(RndA + RndB)>
>UL-C 00 <ek(RndA')> 9000

;Verify the card return value
<SAM 80 7A 00 01 08 <ek(RndA')>
>SAM 9000
```

10.3.3. Example of APDU flow of MIFARE Ultralight C with diversified key

The following authentication step follows the scenario based on Section 9.4. It is assumed that the key inside the MIFARE UL-C card is already diversified.

```
; ACOS6-SAM Card - diversified terminal key local key 2 to RAM with a
diversification data (ie. UL-C's 7-byte UID with a check byte)
<SAM 80 72 03 82 08 <8-byte diversification data>
>SAM 9000

; UL-C Authenticate step 1
<UL-C 1A 00
>UL-C AF <ek(RndB)> 9000

; Prepare Authenticate using terminal key
<SAM 80 78 00 01 08 <ek(RndB)>
>SAM 6110

<SAM 00 C0 00 00 10
>SAM <ek(RndA + RndB')> 9000

; UL-C Authenticate command step 2
<UL-C AF <ek(RndA + RndB)>
>UL-C 00 <ek(RndA')> 9000

;Verify the card return value
<SAM 80 7A 00 01 08 <ek(RndA')>
>SAM 9000
```

11.0. Sample Card Initialization

This section will provide more demonstrations of how to personalize the file header block and create different types of files in a file system. Assuming the card still has no MF, and in Pre-Perso State. The personalization script is set to create the file system in Figure 28:. For purpose of this demonstration, the script will not activate the card to user state and many files have not had the files activated. Therefore, do not treat this file system as complete and secured – all files should have proper access conditions defined and activated, and the card life cycle should be set to user state after initialization.

ACS engineers can help application developers customize a secured file system according to the application's needs. Please contact ACS for further information.

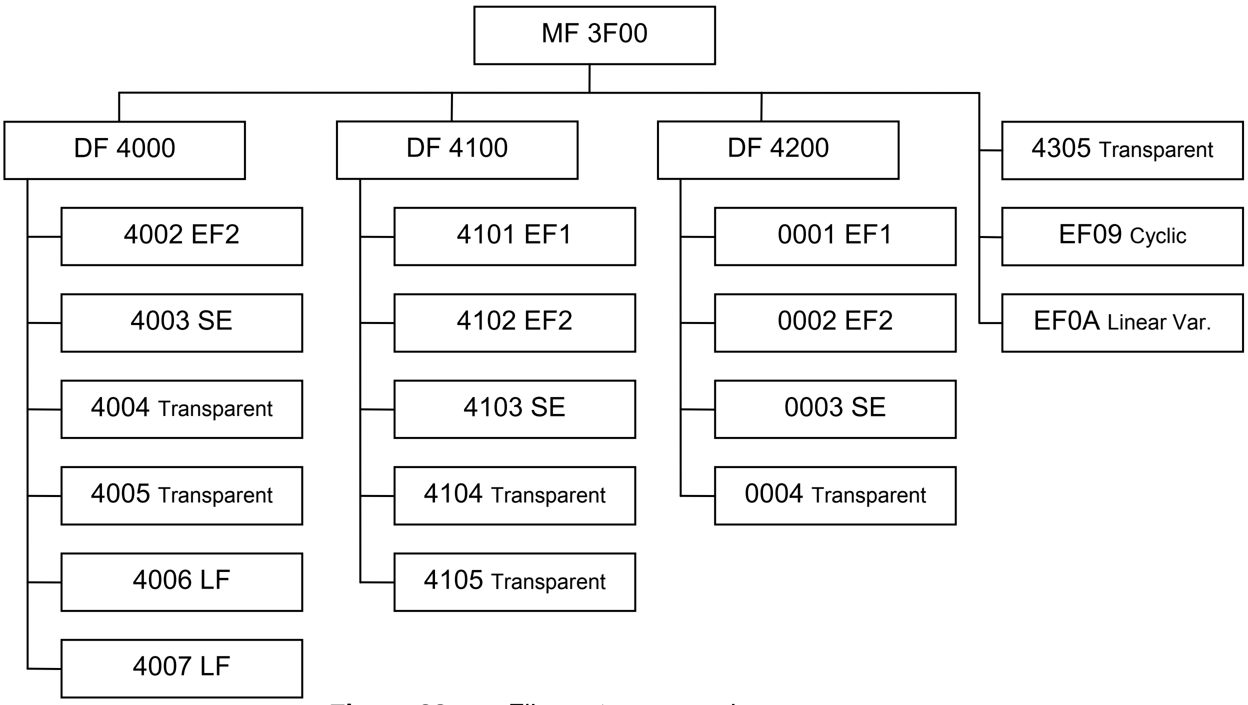


Figure 28: File system example

The script below will use the following color scheme for clarity.
Syntax for ISO-IN: `CLA INS P1 P2 P3 Data (SW1-SW2)`
Syntax for ISO-OUT: `CLA INS P1 P2 P3 [Expected Data Result] (SW1-SW2)`

11.1. Personalization of Card Header

```
; Personalize Card ID Number
00 D0 EE C0 06 01 23 45 (9000)
; Set the Special Function Flags to allow Key Injection Only, Protect Master Key
and Deactivate Card Enable Flags
00 D0 EE F0 01 1F (9000)
```

11.2. Create File System

```
; create MF with DCB = 0xFF
00 E0 00 00 0A 62 08 82 02 3F FF 83 02 3F 00 (9000)

;**********************************************************
; Create DF 4000
; Create DF under MF: ID=4000, DCB=0x00, with SAC and SAE, SE file is 4003
00 E0 00 00 34 62 32 82 01 38 83 02 40 00 84 10 40 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 8A 01 01 8C 08 7F 03 03 03 03 FF FF 03 AB 06 86 02 22 2A 97 00 8D 02 40
03 (9000)

; Under DF 4000, Create KEY FILE EF2: ID=4002, MRL=0x15, NOR=0x04 with SAC (No
access except delete self)
00 E0 00 00 1B 62 19 82 05 0C 01 00 15 04 83 02 40 02 88 01 02 8A 01 01 8C 05 6B 03
FF FF FF FF (9000)

; initialize KEY Records in EF2, follow the KEY data structure
; all KEYS are initially set to 11 22 33 44 55 66 77 88 99 AA BB CC DD EE FF 00
; Keys 1, 2, 3 are external authenticate capable with counter = 0x55 (5 retries)
; while Key 4 is for internal authenticate with usage counter = 0xFFFF (unlimited)
00 DC 00 00 14 81 01 55 00 11 22 33 44 55 66 77 88 99 AA BB CC DD EE FF 00 (9000)
00 DC 00 02 14 82 01 55 00 11 22 33 44 55 66 77 88 99 AA BB CC DD EE FF 00 (9000)
00 DC 00 02 14 83 01 55 00 11 22 33 44 55 66 77 88 99 AA BB CC DD EE FF 00 (9000)
00 DC 00 02 15 84 02 FF FF 00 11 22 33 44 55 66 77 88 99 AA BB CC DD EE FF 00
(9000)

; CREATE SE File: ID=4003, MRL=0x11, NOR=0x04 with SAC (read access = NEVER)
00 E0 00 00 18 62 16 82 05 0C 01 00 11 04 83 02 40 03 8A 01 01 8C 06 6B 03 FF FF FF
FF (9000)

; initialize SE file, follow SE TEMPLATE STRUCTURE
; SE#1: external authentication of local key 1
00 DC 00 00 0B 80 01 01 A4 06 83 01 81 95 01 80 (9000)
;SE#2: external authentication of local key 2
00 DC 00 02 0B 80 01 02 A4 06 83 01 82 95 01 80 (9000)
;SE#3:  external authentication of local key 3
00 DC 00 02 0B 80 01 03 A4 06 83 01 83 95 01 80 (9000)
;SE#4, external authentication of local keys 1 or 2 or 3
00 DC 00 02 11 80 01 04 A4 0C 83 01 81 83 01 82 83 01 83 95 01 80 (9000)

; Create Binary File: ID=4004, SIZE=0096, with SAC
00 E0 00 00 18 62 16 80 02 00 96 82 01 01 83 02 40 04 8A 01 01 8C 06 6E 03 FF FF FF
FF (9000)

; Create Binary File: ID=4005, SIZE=0190, with SAC
00 E0 00 00 18 62 16 80 02 01 90 82 01 01 83 02 40 05 8A 01 01 8C 06 6E 03 FF FF FF
03 (9000)

; Create LF File: ID=4006, MRL=005C, NOR=0A, with SAC
00 E0 00 00 19 62 17 82 05 02 41 00 5C 0A 83 02 40 06 8A 01 01 8C 07 6F 03 FF FF 02
03 04 (9000)

; Create LF FILE: ID=4007, MRL=0024, NOR=0A, with SAC
00 E0 00 00 19 62 17 82 05 02 41 00 24 0A 83 02 40 07 8A 01 01 8C 07 6F 03 FF FF 01
03 04 (9000)

;***********************************************************
; Create DF 4100
; Go back to MF, expect SW1-SW2 to be 61nn
00 A4 00 00 00 (61XX)

; Create next DF: 4100, SE file=4103, with SAC and SAE
00 E0 00 00 43 62 41 82 01 38 83 02 41 00 84 10 41 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 8A 01 01 8C 07 7D 02 02 02 FF FF 02 AB 16 86 02 22 F2 97 00 84 01 22 A0
0B 9E 01 01 A4 06 83 01 81 95 01 08 8D 02 41 03 (9000)

; create PIN FILE EF1 4101: MRL=12h NOR=1, SFI=1
00 E0 00 00 1B 62 19 82 05 0C 01 00 12 01 83 02 41 01 88 01 01 8A 01 01 8C 06 6B FF
FF FF FF FF (9000)

; initialize PIN’s to EF1
; PIN 1: 4 retries left, 4 max retries, PIN = 11 22 33 44 55 66 77 88 99 AA BB CC
DD EE FF 00
```

```
00 E2 00 00 12 81 44 11 22 33 44 55 66 77 88 99 AA BB CC DD EE FF 00 (9000)

; Create KEY FILE EF2 4102, MRL=16, NOR=6, SFI=2
00 E0 00 00 1C 62 1A 82 05 0C 41 00 16 06 83 02 41 02 88 01 02 8A 01 01 8C 07 6F FF
FF FF 02 FF FF (9000)

; initialize KEY RECORDS TO EF2
; all keys are initially set to: 11 22 33 44 55 66 77 88 99 AA BB CC DD EE FF 00
; KEY 1, internal auth., usage counter=0x1234
00 DC 00 00 15 81 02 12 34 00 11 22 33 44 55 66 77 88 99 AA BB CC DD EE FF 00
(9000)
; KEY 2, internal auth., usage counter=0x0000
00 DC 00 02 15 82 02 00 00 00 11 22 33 44 55 66 77 88 99 AA BB CC DD EE FF 00
(9000)
; KEY 3, internal and external auth., usage counter=0x1234, retry counter=0x33 (3
tries)
00 DC 00 02 16 83 03 12 34 33 00 11 22 33 44 55 66 77 88 99 AA BB CC DD EE FF 00
(9000)
; KEY 4, external auth., retry counter=0xFF (unlimited)
00 DC 00 02 14 84 01 FF 00 11 22 33 44 55 66 77 88 99 AA BB CC DD EE FF 00 (9000)
; KEY 5, internal auth., usage counter=0xFF00
00 DC 00 02 15 85 02 FF 00 00 11 22 33 44 55 66 77 88 99 AA BB CC DD EE FF 00
(9000)
; KEY 6, external auth., retry counter=0x88
00 DC 00 02 14 86 01 88 00 11 22 33 44 55 66 77 88 99 AA BB CC DD EE FF 00 (9000)

; Create SE FILE 4103, MRL=27, NOR=05, SFI=3
00 E0 00 00 1B 62 19 82 05 0C 01 00 27 05 83 02 41 03 88 01 03 8A 01 01 8C 06 6B FF
FF FF FF FF (9000)

; initialize RECORD to SE
; SE#1: authenticate local key 3
00 DC 00 00 0B 80 01 01 A4 06 83 01 83 95 01 80 (9000)
; SE#2: authenticate local key 4
00 DC 00 02 0B 80 01 02 A4 06 83 01 84 95 01 80 (9000)
; SE#5; authenticate local key 6
00 DC 00 02 0B 80 01 05 A4 06 84 01 86 95 01 80 (9000)

; Create Transparent File 4104: size=0030, SFI=4
00 E0 00 00 1C 62 1A 80 02 00 30 82 01 01 83 02 41 04 88 01 04 8A 01 01 8C 07 6F FF
FF FF FF FF 01 (9000)

; Create Transparent File 4105, size=0064, SFI=5
00 E0 00 00 1B 62 19 80 02 00 64 82 01 01 83 02 41 05 88 01 05 8A 01 01 8C 06 6E FF
FF FF FF 02 (9000)


;**********************************************************
; Create DF 4200
; go back to MF
00 A4 00 00 00 (61XX)

; create DF ID=4200, DF name='PURSE', SEID = 0003
; SAC - allow all actions if SE#3 is satisfied
; SE file is 0003
00 E0 01 00 43 62 41 82 01 38 83 02 42 00 84 05 50 55 52 53 45 8A 01 01 8D 02 00 03
8C 08 7F 83 83 83 83 83 83 83 AB 1C 85 02 5C 01 9E 01 21 85 02 5C 02 9E 01 22 85 02
56 01 9E 01 21 85 02 56 02 9E 01 22 80 02 06 00 (9000)

; create EF1 ID=0001, MRL=20, NOR=3
; SAC - allow all actions if SE#3 is satisfied
00 E0 01 00 1D 62 1B 82 05 0C 01 00 20 03 83 02 00 01 88 01 01 8A 01 01 8C 08 7F 83
83 83 83 83 83 83 (9000)
; APPEND RECORDS TO EF1
; PIN1 and PIN2: 3 tries
00 E2 00 00 12 81 33 11 22 33 44 55 66 77 88 99 AA BB CC DD EE FF 00 (9000)
00 E2 00 00 12 82 33 11 22 33 44 55 66 77 88 99 AA BB CC DD EE FF 00 (9000)
```

```
00 E2 00 00 12 83 33 11 22 33 44 55 66 77 88 99 AA BB CC DD EE FF 00 (9000)

; create EF2 ID=0002, MRL=20, NOR=5
; SAC - allow all actions if SE#3 is satisfied
00 E0 01 00 1D 62 1B 82 05 0C 02 00 20 05 83 02 00 02 88 01 02 8A 01 01 8C 08 7F 83
83 83 83 83 83 83 (9000)
; APPEND KEYS TO EF2
; KEY 1, 5 tries, 20 usage, int. ext. capable
00 E2 00 00 16 81 03 00 20 55 00 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11
(9000)
; KEY 2, 5 tries, 20 usage, int. ext. capable
00 E2 00 00 16 82 03 00 20 55 00 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22
(9000)
; KEY 3, 5 tries, 20 usage, int. ext. capable
00 E2 00 00 16 83 03 00 20 55 00 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33
(9000)
; KEY 4 (derived), 5 tries, 20 usage, int. ext. capable
00 E2 00 00 16 84 03 00 20 55 00 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44
(9000)

; create SE EF ID=0003, MRL=20, NOR=8
; SAC - allow all actions if SE#3 is satisfied
00 E0 01 00 1D 62 1B 82 05 0C 03 00 20 08 83 02 00 03 88 01 03 8A 01 01 8C 08 7F 83
83 83 83 83 83 83 (9000)
; APPEND RECORD to SE
; SE#1: submit PIN1
00 E2 00 00 0B A4 06 83 01 81 95 01 08 80 01 01 (9000)
; SE#2: submit PIN2
00 E2 00 00 0B A4 06 83 01 82 95 01 08 80 01 02 (9000)
; SE#3: submit key 1, key 2, key 3
00 E2 00 00 11 A4 0C 83 01 81 83 01 82 83 01 83 95 01 08 80 01 03 (9000)

; create binary EF ID=0004, size = 0080
; SAC - allow U/W if SE#3 is satisfied
00 E0 01 00 18 62 16 80 02 00 80 82 01 01 83 02 00 04 88 01 04 8A 01 01 8C 03 06 23
23 (9000)

; Activate DF 4200
00 44 00 00 02 00 01 (9000)
00 44 00 00 02 00 02 (9000)
00 44 00 00 02 00 03 (9000)
00 44 00 00 02 00 04 (9000)
00 44 00 00 00 (9000)


;**********************************************************
; Create MF level EFs.
; go back to MF
00 A4 00 00 00 (61XX)

; create DATA FILE 4305, SFI=5
00 E0 00 00 1B 62 19 80 02 08 00 82 01 01 83 02 43 05 88 01 05 8A 01 01 8C 06 6E FF
FF FF 01 01 (9000)
; Creating a Cyclic File with file ID=EF09, MRL=0A, NOR=03, R/W access allowed if
SE#1 is satisfied
00 E0 00 00 12 62 10 83 02 EF 09 82 05 06 00 00 0A 03 8C 03 03 81 81 (9000)
; create Linear Varible EF0A, MRL=10, NOR=01, R/W access allowed if SE#1 is
satisfied
00 E0 00 00 12 62 10 83 02 EF 0A 82 05 04 00 00 0A 01 8c 03 03 81 81 (9000)
```

11.3. Demonstrating File Behaviors

```
; Demonstrating transparent file behavior
; Select Transparent data file 4305
00 A4 00 00 02 43 05 (611E)
```

```
; test TRANSPARENT file commands
; invalid P1/P2
00 B0 FF FF 00 (6B00)
00 B0 FF 00 00 (6B00)


; read 0x10 bytes, starting from offset 0000
00 B0 00 00 10 [FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF] (9000)

; update binary at offset 0x00, 0x20 bytes
00 D6 00 00 20 11 22 33 44 55 66 77 88 99 AA BB CC DD EE FF 00 11 22 33 44 55 66 77
88 99 AA BB CC DD EE FF 00 (9000)

; update binary at offset 0x0100, 9 bytes
00 D6 01 00 09 11 22 33 44 55 66 77 88 99 (9000)
00 D6 07 80 08 11 22 33 44 55 66 77 88 (9000)
00 D6 07 F8 08 88 77 66 55 44 33 22 11 (9000)

; read binary, check if the data written are correct
00 B0 00 00 20 [11 22 33 44 55 66 77 88 99 AA BB CC DD EE FF 00 11 22 33 44 55 66
77 88 99 AA BB CC DD EE FF 00] (9000)
00 B0 01 00 09 [11 22 33 44 55 66 77 88 99] (9000)
00 B0 07 F8 08 [88 77 66 55 44 33 22 11] (9000)

;**********************************************************
; Demonstrating Cycle file behavior
; Select cycle file EF09
00 A4 00 00 02 EF 09 (611B)

; if we write 3 records
00 DC 00 00 0A 11 11 11 11 11 11 11 11 11 11 (9000)
00 DC 00 02 0A 22 22 22 22 22 22 22 22 22 22 (9000)
00 DC 00 02 0A 33 33 33 33 33 33 33 33 33 33 (9000)
; the 1st record is the last record written
00 B2 00 00 0A [33 33 33 33 33 33 33 33 33 33] (9000)
; reading the next record will wrap to file forward
00 B2 00 02 0A [11 11 11 11 11 11 11 11 11 11] (9000)
; reading the previous record will wrap the file back
00 B2 00 03 0A [33 33 33 33 33 33 33 33 33 33] (9000)
00 B2 00 03 0A [22 22 22 22 22 22 22 22 22 22] (9000)

;**********************************************************
; Demonstrating linear variable file behavior
; Select linear variable file EF0A
00 A4 00 00 02 EF 0A (611B)

; write and read the record
00 DC 00 00 0A AA AA AA AA AA AA AA AA AA AA (9000)
00 B2 00 00 0A [AA AA AA AA AA AA AA AA AA AA] (9000)
; write again
00 DC 00 00 03 11 22 33 (9000)
; read back the whole record. Unlike LF file, the whole record is erased first
before writing 11 22 33
00 B2 00 00 0A [11 22 33 FF FF FF FF FF FF FF] (9000)
```