Advanced Card Systems Ltd.
Card & Reader Technologies

ACOS6-SAM (Contact)

Reference Manual V2.28

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

Document Revision History

Date	Description	Version
2012-06-21	• Added MIFARE DESFire and MIFARE Plus card support • Editing corrections • Formatting corrections	2.17
2012-07-05	• Formatting corrections	2.18
2012-07-13	• Formatting corrections	2.19
2012-07-24	• Editing corrections	2.20
2012-08-06	• Editing corrections	2.21
2012-08-15	• Editing corrections	2.22
2013-02-18	• Formatting corrections	2.23
2014-10-21	• Editing corrections	2.24
2015-01-05	• Added Customization of ATR for Null Driver • Added Automatic download of Null Driver from Windows Update	2.25
2017-02-16	• Updated memory from 32KB to 64KB	2.26
2019-05-20	• Added AES • Added DESFire EV2 • Added DESFire Light • Updated Section 1 and 2	2.27
2019-12-19	• Removed diagram in Section 3 • Editing corrections	2.28

Table of Contents

List of Figures

List of Tables

1.0. Introduction

This document aims to provide a detailed description of the features and functions of the ACS Smart Card Operating System Version 6 – Security Access Module, also known as ACOS6-SAM, developed by Advanced Card Systems Ltd.

1.1. History of Modification

Date	Version and Description
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 KB • Added bulk encryption with CBC
November 2008	**ACOS6-SAM revision 4.02** • Expanded user storage capacity to 32 KB • Added FIPS 140-2 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=95h for increased compatibility
May 2009	**ACOS6-SAM revision 4.06** • Expanded user storage capacity to 64 KB • Multiple purse file can be used in one DF
July 2009	**ACOS6-SAM revision 4.07** • Supports MIFARE Ultralight C diversification and authentication.
December 2010	**ACOS6-SAM revision 4.08** • Support AES-128 Bulk Encryption • 32KB EEPROM • MIFARE DESFire/MIFARE DESFire EV1 Security Support • Mutual Authenticate between ACOS6-SAM and MIFARE DESFire • MIFARE DESFire Card Session Key store in SAM Card • Key Injection from ACOS6-SAM to MIFARE DESFire • MAC and CMAC capabilities • Fully-enciphered communication with MIFARE DESFire

Date	Version and Description
May 2012	**ACOS6-SAM revision 4.09** • Support MIFARE Plus diversification, authentication, key injection and secure communications
June 2016	**ACOS6-SAM revision 4.17** • Support ACOS3 using 3Key 3DES / AES-128bit / AES-192bit • Expanded user storage capacity to 64 KB
April 2019	**ACOS6-SAM revision 4.20** • MIFARE DESFire EV2 Support • MIFARE DESFire Light Support

Table 1: History of Modification – ACOS6-SAM

1.2. Symbols and Abbreviations

Abbreviation	Description
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(C, 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

Abbreviation	Description
LSB	Least Significant Byte
MAC	Message Authentication Code
MF	Master File
MFP	MIFARE Plus
MOC	Ministry of Construction – China specifications
MSb	Most Significant Bit
MSB	Most Significant Byte
Nibble	four- bit aggregation; a byte consists of two nibbles
PBOC	People's Bank of China specifications
RFU	Reserved for Future Use
RMAC	Retail MAC
SL0	MIFARE Plus Security Level 0
SL1	MIFARE Plus Security Level 1
SL2	MIFARE Plus Security Level 2
SL3	MIFARE Plus Security Level 3
SAC	Security Attribute – Compact
SAE	Security Attribute – Expanded
SAM	Secure Access 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
TTREFC	Terminal Transaction Reference for Credit
TTREFD	Terminal Transaction Reference for Debit
UQB	Usage Qualifier Byte
II	Concatenation

Table 2: Symbols and Abbreviations

1.3. Organization

This document is arranged in the following manner:

Section 3.0– General overview of SAM functionality and application usage. It discusses the specialized SAM commands that this card supports.

Section 4.0 – Basic card management. The pre-customization of the COS and changing of basic card features including ATR, life cycle, etc.

Section 5.0 – Card's file system. The card headers and what is contained in the internal security and purse files.

Section 6.0 – Security features of the card. The card security options, including file security, mutual authentication, secure messaging and secure PIN submissions.

Section 7.0 – Command reference of ACOS6-SAM commands.

Section 8.0 – Command reference of SAM specific commands.

Section 9.0 – Reference of all ACOS6-SAM Status Codes.

Section 10.0 – Different command flows and interaction with an ACOS3 client card.

Section 11.0 – Different command flows and interactions with different MIFARE products.

Section 12.0 – Quick start guide and examples when using the ACOS6-SAM as a SAM card.

Section 13.0 – Examples of file system creation and how to personalize and program the ACOS6-SAM. This includes creating different types of files, adding security files, etc.

2.0. Technical Specifications

The following are some of the technical properties of the ACOS6-SAM card:

2.1. Electrical

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

2.2. Environmental

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

2.3. Communication Protocols

- T=0 with baud up to 223,200 bps

2.4. Memory

- Capacity: 64 KB
- EEPROM endurance: 100,000 erase/write cycles
- Data retention: 10 years

2.5. Cryptographic Capabilities

ACOS6-SAM supports a number of cryptographic algorithms, including:

- DES, 2K3DES, 3K3DES (ECB, CBC)
- AES: 128/192 bits (ECB, CBC)

2.6. File Security

- FIPS 140-2 compliant hardware-based random number generator
- Session key based on random numbers
- Key pair for mutual authentication
- Secure Messaging function for confidential and authenticated data transfers
- Stores and performs all key operations for mutual authentication, encrypted PIN submission, secure messaging, and e-Purse commands
- Multilevel secured access hierarchy
- Anti-tearing capability

2.7. Secure Access Module Compatibility

The ACOS6-SAM pairs with the following client cards:

- ACOS3
- ACOS6
- ACOS7
- ACOS10

- MIFARE Ultralight® C
- MIFARE® DESFire®
- MIFARE® DESFire® EV1
- MIFARE® DESFire® EV2
- MIFARE® DESFire® Light
- MIFARE Plus®

2.8. Answer to Reset (ATR)

After a hardware reset (e.g. power up), the card transmits an **Answer To Reset (ATR)** in compliance with ISO 7816 Part 3. ACOS6-SAM supports the protocol type T=0 in direct convention.

The following is the default ATR:

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

Table 3: Configuration of the Answer-to-Reset

The ATR may be completely changed using the ATR file. See **Answer To Reset (ATR)** and **Customizing the ATR** for more information.

2.9. Compliance to Standards

- Compliant with ISO 7816 Parts 1, 2, 3, 4

3.0. ACOS6-SAM Security Features

The ACOS6-SAM is designed to be used as a general cryptogram computation module or as the security authentication module for client cards such as ACOS3, ACOS6, ACOS7, ACOS10, MIFARE Ultralight C, MIFARE DESFire, MIFARE DESFire EV1, MIFARE DESFire EV2, MIFARE DESFire Light and MIFARE Plus 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 leave the SAM un-encrypted and the system security is greatly enhanced.

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 ACOS3 or ACOS6 client smart cards, ACOS6-SAM can perform the following functions:

- Initialize the ACOS3/6 card with diversified keys based on the card's unique serial number
- Perform mutual authentication process, and generate session key
- Perform secure messaging with ACOS3/6
- Compute secured e-Purse commands
- Perform secure key injection with ACOS6

When used with ACOS7, ACOS10, and ACOS10-PSAM smart cards, ACOS6-SAM can perform the following functions:

- Initialize the ACOS7/10/10-PSAM card with diversified keys based on the card's unique serial number
- Perform mutual authentication process and generate session key
- Perform secure key injection with ACOS7/10

  Note: *ACOS6-SAM does not perform PBOC/MOC-based purse commands. For that, please use ACOS10-PSAM which is a PBOC-compliant payment SAM.*

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

- Initialize the UL-C client card with diversified keys based on the card's unique serial number
- Perform mutual authentication process

When used with MIFARE Plus, MIFARE DESFire/MIFARE DESFire EV1/EV2/Light smart cards, ACOS6-SAM can perform the following functions:

- Initialize the MIFARE Plus, MIFARE DESFire/MIFARE DESFire EV1/EV2/Light client card with diversified keys based on the card's unique serial number
- Perform mutual authentication process
- Perform secure messaging
- Perform secure key injection

The programming method of ACOS6-SAM is different from ACOS3 cards. It is designed to conform to the ISO 7816 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.

3.1. Generate key

This command generates a diversified key based on a Master Key already residing 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.

3.2. Diversify key

This command computes the ACOS3/6/7/10, UL-C, DESFire, DESFire EV1/EV2/Light or MIFARE Plus 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 for bulk encryption.

Note: *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.*

3.3. Encrypt

This command performs and returns Single/Triple DES encryption in ECB, CBC or MAC modes using a 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 a normal ACOS3/6, DESFire, DESFire EV1/EV2/Light or MIFARE Plus command to its secure message counterpart after authentication with the corresponding card. The SAM card can perform command data encryption and command MAC in one step for the client card SM command. The session key and sequence number will be kept and updated in the SAM card.

3.4. 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, DESFire, DESFire EV1/EV2/Light or MIFARE Plus secure messaging MAC and decryption of the encrypted response data. It should be called after an ENCRYPT command of SM computation mode. The session key and sequence number will be kept and updated in the SAM card.

3.5. Prepare authentication

This command helps the application perform Mutual Authentication with an ACOS3/6/7/10, MIFARE UL-C, DESFire, DESFire EV1/EV2/Light or MIFARE Plus 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 ACOS3/6/7/10, MIFARE UL-C, DESFire, DESFire EV1/EV2/Light or MIFARE Plus challenge data to the SAM, and this command will return the required terminal

authentication data required by the client card, 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 the client ACOS card and ACOS6-SAM.

3.6. Verify authentication

This command completes Mutual Authentication with an ACOS3/6/7/10, MIFARE UL-C, MIFARE DESFire, MIFARE DESFire EV1/EV2/Light or MIFARE Plus client card. A successful call to the *Prepare Authentication* command is needed before using this command.

After a successful execution of an AUTHENTICATE (and GET REPONSE) command with the client card, the card authentication data from the client card is sent to the ACOS6-SAM card where it will check if the session key generated is correct and complete the mutual authentication process

3.7. Verify ACOS inquire account

This command helps the application verify the MAC returned by the 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.

3.8. Prepare ACOS account transaction

This command helps the application perform PURSE commands (CREDIT, DEBIT) with an ACOS3/6 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.

3.9. Verify ACOS debit certificate

This command helps the application verify the DEBIT CERTIFICATE returned by the 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.

4.0. Card Management

This section outlines the card level features and management functions.

4.1. Card Life Cycle States

ACOS6-SAM has the following card states:

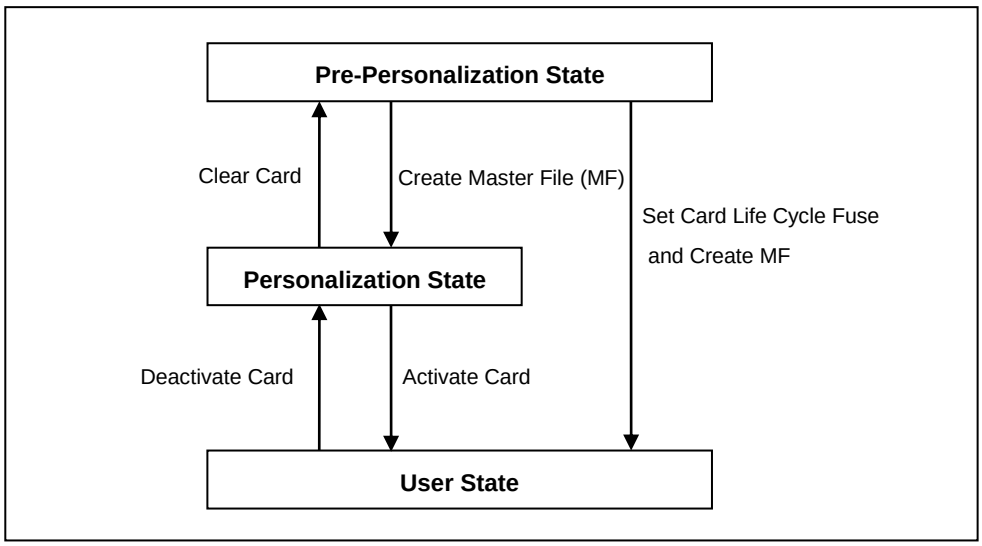


Figure 1: Card Life Cycle States

4.1.1. Pre-Personalization State

This is the initial state of the card. The user is allowed to freely access the card header block (defined in **Card Header Block**). The card header block can be referenced by its address using the READ BINARY or UPDATE BINARY command.

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

4.1.2. Personalization State

The card goes into this state once the MF is successfully created and the *Card Life Cycle Fuse* is not blown (still FFh). The user can no longer directly access the card's memory as in the previous state. The 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.

4.1.3. User State

The card goes into this state once the MF is successfully created and the *Card Life Cycle Fuse* is blown. Alternatively, users can use the ACTIVATE CARD command to go from the personalization state to the user state.

The card cannot revert back to previous states when the *Card Life Cycle Fuse* is set (00h) 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 in this state.

4.1.4. Typical Development Steps of a Card

1. The user personalizes the card's header block using UPDATE BINARY.
2. The user then creates his card file structure, starting with the MF. Dedicate Files (DF) and Elementary Files (EF) 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, the user performs CLEAR CARD command and blows the *Card Life Cycle Fuse* using the UPDATE BINARY command (write 00h to address EEC7h).
4. The card goes into Operational Mode, when the MF is created again. The user can re-construct the file system under this state. At this point, the card can no longer go back to previous states.

The user may choose to set the enable DEACTIVATE CARD command in the 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.

4.2. Card Header Block

ACOS6-SAM is a card operating system that has 64 KB EEPROM. In its initial state (where no file exists), the 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. The 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 the Card Serial number, which is a unique serial number per card already available in ACOS6-SAM (See **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 (FFh), the card OS will allow the user to erase the whole card's contents. At such state, the user is allowed to re-program the card's header block. See **Card Life Cycle States** for more details.

EEC8 - **EECA**: Random Number Counter – Deprecated

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

EECB: ACOS3 Record Numbering Mode – Deprecated

For compatibility purposes, this field serves as the RECORD_NUMBERING_FLAG in ACOS3 (in file FF01h). 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 flags 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 4: Special Function Flags

Key Injection Only Flag — After setting this bit to zero and activating the Key file, the card must use *Get Key* and *Set Key* commands to access the 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 FFh.

4.3. Answer To Reset (ATR)

After a hardware reset (e.g. power up), the card transmits an **Answer To Reset (ATR)** in compliance with ISO 7816 Part 3. ACOS6-SAM supports the protocol type T=0 in direct convention.

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

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

Table 5: Default Configuration of the Answer to Reset

The 14 historical characters are composed of the following:

Historical Characters	ATR	Description
T1	41h	Indicates ACOS Card
T2	03h	Major version
T3	00h	Minor version
T4	00h	Not used
T5	00h	
T6	00h	
T7	00h	
T8	00h	
T9	00h	
T10	00h	
T11	00h	
T12	XX	Card Life Cycle State indicator: 02h: Perso (or pre-perso) State 00h: User State
T13	90h	Not used
T14	00h	

Table 6: Answer to Reset Historical Bytes

4.3.1. Customizing the ATR

ACOS6-SAM's ATR can have a customized 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 recommended to only change the 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: *The actual baud rate will depend on the card reader's capability and its oscillator. Although ACS has tested the card on all TA1 values with the major readers on the market, some readers may have non-standard timing and may not support the highest speed offered by ACOS6-SAM.*

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

Parameter	Recommended ATR Value	Description
TS	3Bh	Direct Convention. ***Important:*** *keep this value.*
T0	Bxh	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	 96h 19h 18h 95h 94h 93h 92h 11h	The following are the reference baud rate[2] and their values: 223,200 bps 192,000 bps 115,200 bps 111,600 bps 55,800 bps 27,900 bps 13,950 bps 9,600 bps
TB1	00h	No programming voltage required.
TD1	00h	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.		

Table 7: Customization of the Answer-to-Reset

Notes:

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

Example:

To change the card's ATR to ACOS3, 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
```

4.3.1.1. Customized ATR for Microsoft Windows Usage

For Windows® 7 and above operating systems: Windows automatically attempts to download the smart card's minidriver whenever a smart card is inserted into the smart card reader. Since ACOS6-SAM is not intended to conform to Windows default usage, such smart card minidriver is not necessary. However, if the ACOS6-SAM is inserted into a Windows system, Windows may search online for the driver and may give a warning that the "device driver was not successfully installed" for the smart card. There are two ways to solve this issue:

1. Disable smart card plug and play and certificate propagation in Windows.
2. Change the ATR so Windows will recognize the ACOS6-SAM smart card to use ACS's Unified Null Driver.

For the first solution, please follow the instructions in this Microsoft® support link to disable smart card plug and play: http://support.microsoft.com/kb/976832. This may have to be done for every computer that will be used in this system..

For the second solution, ACS has developed a Unified Null driver for the ACOS line of smart cards. The Unified Null driver will satisfy the Windows requirement to have a minidriver for the card, hence the warning from Windows every time the card is inserted will no longer appear. The Unified Null Driver can be downloaded automatically from Windows Update if Automatic Updates are turned ON. In order for Windows to recognize the ACOS6-SAM smart card to use the Unified Null driver, the ATR must be customized. In the case of ACOS6-SAM, the ATR should be:

3B BE XX 00 00 41 43 4F 53 36 5F 53 41 4D 5F 4E 44 90 00h.

The XX is the value of TA1. The TA1 value can be set to the baud rate that the smart card reader used can support. For ACS ACR38 for example, the TA1 value can be set to TA1=95h.

To change the ATR, perform the following:

1. Write the ATR Length: 00 D6 EE C6 01 13h.
2. Write the Customized ATR for ACOS6-SAM to the ATR Historical Bytes: 00 D6 EE D0 13 3B BE 95 00 00 41 43 4F 53 36 5F 53 41 4D 5F 4E 44 90 00h.
3. Reset the card. ATR is now 3B BE 95 00 00 41 43 4F 53 36 5F 53 41 4D 5F 4E 44 90 00h.

   ***Note**: The ATR must be customized before the MF is created.*

5.0. File System

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

5.1. 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 (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 the figure below:

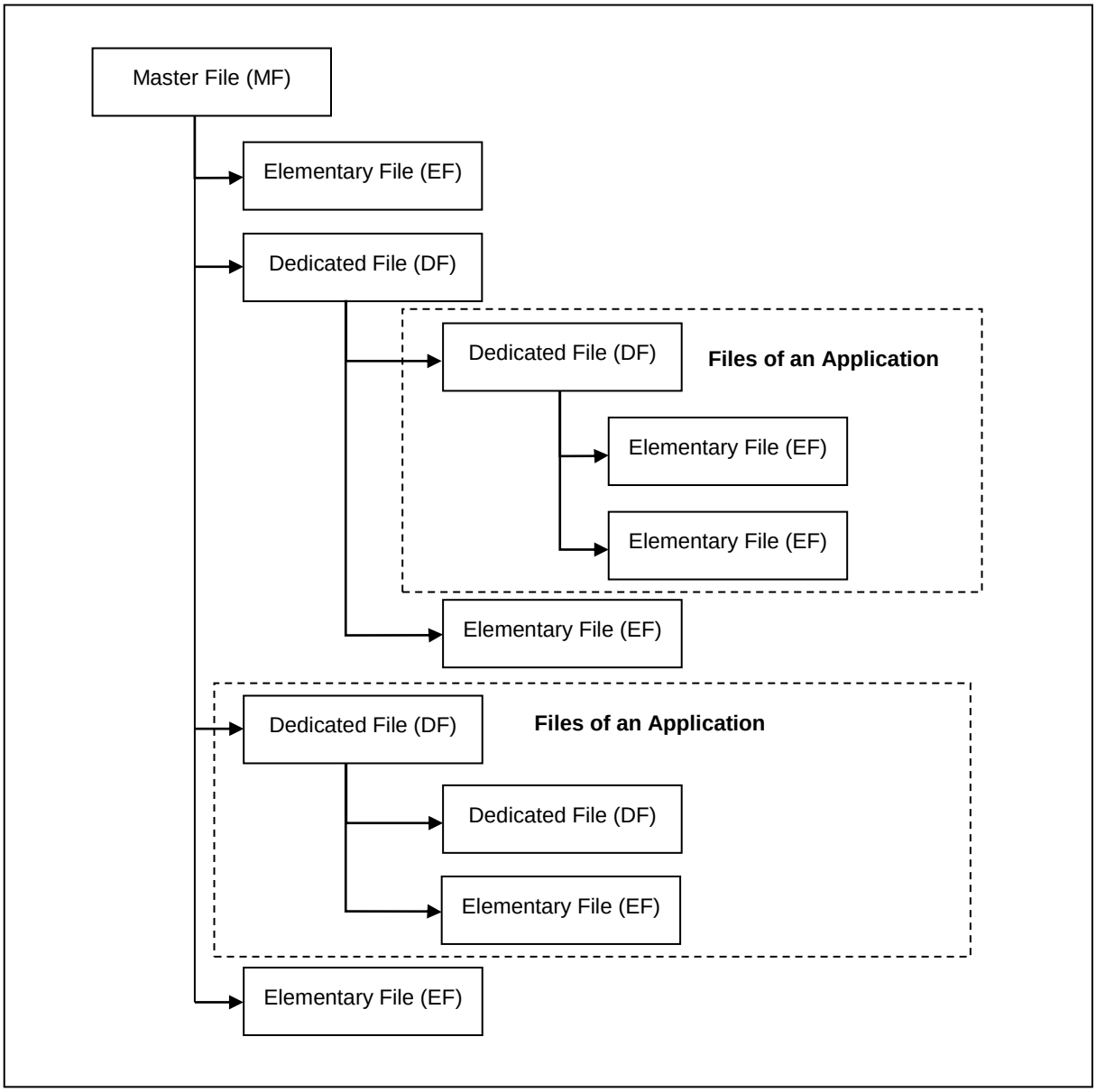


Figure 2: Sample of File System Hierarchy

5.2. 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 8: File Header Block

5.2.1. 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	3Fh	**MF**
0	0	1	1	1	0	0	0	38h	**DF**
0	0	0	0	0	-	-	-	-	**Working EF**
0	0	0	0	0	0	0	1	01h	Transparent (binary) EF
0	0	0	0	0	0	1	0	02h	Linear Fixed EF
0	0	0	0	0	1	0	0	04h	Linear Variable EF
0	0	0	0	0	1	1	0	06h	Cyclic EF

b7	b6	b5	b4	b3	b2	b1	b0	Hex	File type
0	0	0	0	1	-	-	-	-	**Internal EF**
0	0	0	0	1	1	0	0	0Ch	Internal Linear Variable EF (or KEY EF)
0	0	0	0	1	1	1	0	0Eh	Internal Cyclic EF (Purse EF)

Table 9: File Descriptor Byte

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

5.2.2. Data Coded Byte (DCB)

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

5.2.3. 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 the MF) must be unique. There are a few pre-defined File IDs. They are:

3F00h - Master File

3FFFh - Current DF

FFFFh - RFU

A file cannot have an ID of 3FFFh and FFFFh.

5.2.4. 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 EFs, the first byte indicates the *maximum record length* (MRL), while the second indicates the *number of records* (NOR). For non-record-based EFs (Transparent EF), the first byte represents the high byte of the file size and the second is the low-order byte. For DF's, this field is not used.

5.2.5. Short File Identifier (SFI)

The *Short File Identifier* is a five-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. Two files may have the same SFI under a DF. In such case, ACOS6-SAM will select the one created first.

5.2.6. Life Cycle Status Integer (LCSI)

This byte indicates the life status of the file, as defined in ISO 7816 Part 4. It can have the following values:

b7	b6	b5	b4	b3	b2	b1	b0	Hex	Meaning
0	0	0	0	0	0	0	1	01h	Creation state
0	0	0	0	0	0	1	1	03h	Initialization state
0	0	0	0	0	1	-	1	05h or 07h	Operational state (activated)
0	0	0	0	0	1	-	0	04h or 06h	Operational state (deactivated)
0	0	0	0	1	1	-	-	0Ch to 0F h	Termination state

Table 10: Cycle Status Byte

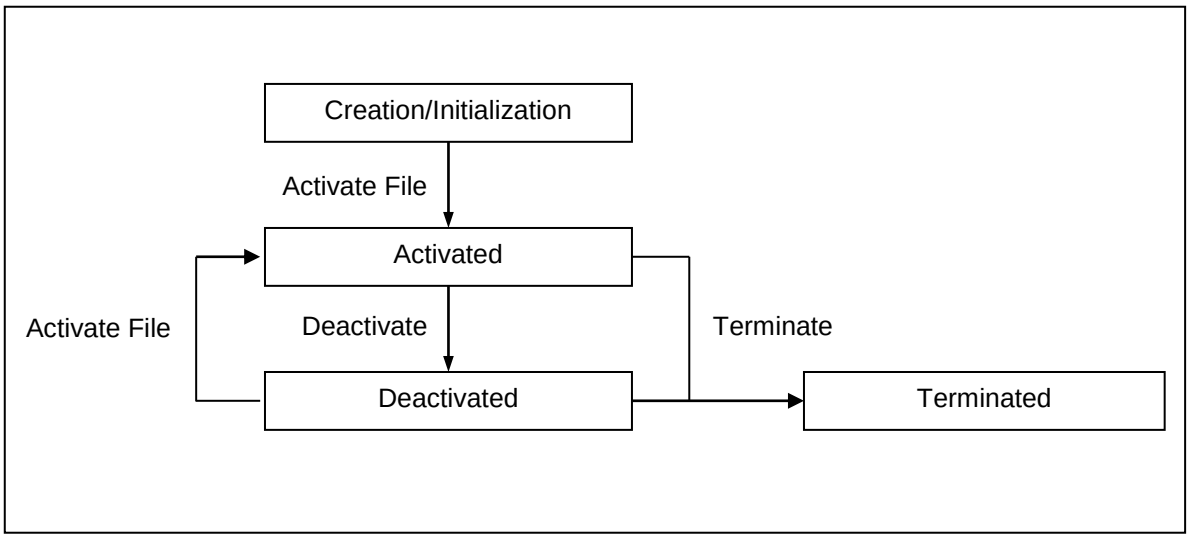


Figure 3: Life Cycle Status Integer

1. In Creation/Initialization state, all commands to the file will be allowed by the COS.
2. In Activated state, commands to the file are allowed only if the file's security conditions are met.
3. In Deactivated state, most commands to the file are not allowed by the COS.
4. In Terminated State, all commands to the file will not be allowed by the COS.

5.2.7. Security Attribute Compact Length (SAC Len)

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

5.2.8. Security Attribute Expanded Length (SAE Len)

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

5.2.9. 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.

5.2.10. Parent Address

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

5.2.11. 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.

5.2.12. 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 **Security Attribute Compact (SAC)** for more information.

5.2.13. 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 **Security Attribute Expanded (SAE)** for more information.

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

5.2.14. SE File ID (for DF only)

For a DF, this field consists of two 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.

5.2.15. FCI File ID (for DF only)

For a DF, this field consists of two 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.

5.2.16. DF Name (for DF only)

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

5.3. 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 must be set to NEVER. Typically, a DF should have a:

1. Key File to hold PIN codes (referred as EF1) for verification.
2. Key File to hold KEY codes (referred as EF2) for authentication.
3. SE File to hold security conditions.

A Key File is an Internal Linear Variable file. It may contain a PIN data structure or a KEY data structure.

5.3.1. PIN Data Structure

The PIN is used for the VERIFY command. The file that contains the 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)

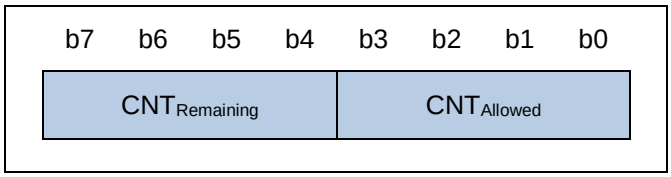
Figure 4: PIN Data Structure

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 11: 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 ($CNT_{Remaining}$).



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 FFh, then the error counter is not used and the PIN allows for an unlimited number of retries. Hence, the maximum number of retries short of unlimited is 14 or 0Eh.

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

5.3.2. 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 or 24

Figure 5: 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 – 0000b
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 12: Key Type Byte

Internal authentication keys are used for the terminal to authenticate the card. It can also be used by the 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 the 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 a secured encryption engine. But it may not be desirable if that key is used for the master key.

Key Info: This field depends on the Key Type field. It contains the 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 generating a 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 (FFFEh). If the counter is FFFFh, then the usage of the key is unlimited.

The **external authentication error counter** is the same as the PIN Error Counter. Please see **PIN Data Structure** 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 a retry counter are in the key info. When either the usage counter or the retry counter reach zero, the key will be rendered invalid. If the 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
01h, 05h, 08h, 09h, 0Ch, 0Dh	1 byte error counter
02h, 06h	2 byte usage counter
03h, 07h, 0Ah, 0Bh, 0Eh, 0Fh	2 byte usage counter followed by 1 byte error counter
04h	No key info byte
All other values	RFU

Table 13: Relationship between Key Type and Key Info Bytes

Algorithm Reference States the cryptographic algorithm allowed to be used for this key.

Bit	Description
00h	3DES
01h	DES
02h	3 Key DES (For SAM command Use only)
03h	AES-128 /AES-192 (For SAM command Use only)
Other	RFU

Key The single, 16-byte triple DES, and 24-byte triple DES key must have lengths of 8, 16, and 24 bytes respectively.

5.3.3. 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 **Security Environment** for more information.

6.0. Security Features

This chapter illustrates the access rights and security capabilities of the ACOS6-SAM card along with its environment and usage.

Furthermore, 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 PINs or Keys are submitted or authenticated.

Global PINs 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.

6.1. 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)*.

6.1.1. Security Attribute 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 the 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 14: 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 **Secure Messaging for Authenticity (SM-MAC)**

b7	b6	b5	b4	b3	b2	b1	b0	Meaning
-	1	-	1	-	-	-	-	ISO Secure messaging (Encryption and MAC – according to **Secure Messaging for Confidentiality (SM-ENC)**

Table 15: 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 02h means:

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

SC: 02 03 04 FF FF 02h

- 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

6.1.2. Security Attribute Expanded (SAE)

The SAE is a data structure that resides in each file. It tells the COS whether to allow file commands to proceed or not. 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 16: Access Mode Data Object (AMDO)

SCDO May have the following Tags (and Length):

Tag	Length	Value	Meaning
90h	00h	-	Allow
97h	00h	-	Never
9Eh	01h	SC Byte	SC Byte the same as SAC

Tag	Length	Value	Meaning
A4h	Var.	Authentication Template	Allow if AT condition is satisfied
A0h	Var.	SC DO	One or more SC DO follows where at least 1 condition is met
AFh	Var.	SC DO	One or more SC DO follows where all conditions are met

Table 17: Security Condition Data Object (SCDO)

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

<AMDO>: 86 02 22 F2h -> command with INS=22h and P1 =F2h

<SCDO>: 97 00h -> never

Hence, the SAE tells the COS not to allow APDU commands with INS=22h and P1=F2h.

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

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

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

Hence, the SAE tells the COS to allow commands with INS=22h 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.

6.2. 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 80h with the length of 01h.

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 A4h and its value contains one or more data objects.

Tags recognized within AT:

Tag	Length	Meaning
83h	01h	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.
95h	01h	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 18: 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 80h

It has the following meaning:

- The SE ID is 03h (SE#3).
- AT is: A4 09 83 01 84 83 01 81 95 01 80h; this contains two 83 tags inside. This means:
  - Allow command if local KEY 84 is authenticated;
  - Allow command if local KEY 81 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 88h

It has the following meaning:

- The SE ID is 05 (SE#5).
- AT is A4 09 83 01 84 83 01 81 95 01 88h; contains two conditions inside it:
- First condition in AT is: 83 01 84 95 01 88h -> allow command if local KEY 84h is authenticated AND if local PIN 84h is verified;
- Second condition in AT is: 83 01 01 95 01 88h -> allow command if global KEY 01h is authenticated AND if global PIN 01h 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.

6.3. 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 between 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 ACOS3. Hence, it is backward compatible. As in ACOS3, the mutual authentication procedure involves two main commands, GET CHALLENGE and MUTUAL AUTHENTICATE. GET RESPONSE is also necessary after the MUTUAL AUTHENTICATE command.

6.3.1. 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 to the GET CHALLENGE command.

Please follow the figure in the next page for the detailed steps of the mutual authentication procedure.

Terminal		ACOS6-SAM
GET CHALLENGE 00 84 00 00 08h	➔	
	➔	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 82h 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

Figure 6: Mutual Authentication Procedure

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

6.3.2. Session Key Computation

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 an 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:

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

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

$KsL =$ 3DES (3DES (RND_C, K_C), K_T)

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

$KsR =$ 3DES (RND_T, REV (K_T))

The variables are the same as that in the **Mutual Authentication Procedure**. 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, FFh will be padded)

K_T: 16-byte Terminal Key (if key length < 16, FFh will be padded)

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

B7 B6 B5 B4 B3 B2 B1 B0h → B3 B2 B1 B0 B7 B6 B5 B4h

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

$KS =$ DES (DES (RND_C, K_C) XOR RND_T, K_T)

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)

6.4. 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 a one-time-password that is more optimal for human input.

Terminal		ACOS6
GET CHALLENGE 00 84 00 00 04h	➔	Generate and return card challenge RNDC
	➔	
1. Compute R = Left 4 Byte of 3DES (00 00 00 00 || RND_C, K_T) 2. Send terminal response and card challenge EXTERNAL AUTHENTICATE (R) 00 82 00h K_T 04h 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 00h || RND_C, K_T)
	➔	3. Turn on K_T Access Right Return 90 00h

Figure 7: Short Key External Authenticate Procedure

6.5. 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 **Secure Messaging for Confidentiality (SM-ENC)** discusses SM for confidentiality.

SM for Authentication allows data and commands that are transferred into the card and vice versa to be authenticated. This ensures 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 **Mutual Authentication**.

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 Activate File Deactivate File Terminate DF Terminate EF Delete File	Read Binary Read Record

Table 19: Secure Messaging Commands

6.5.1. 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 04h. 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 **Computing the Secure Messaging Retail MAC (RMAC)**. 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 an 80h byte followed by 0 to 7 00h to make the length of the data to be authenticated equal to a multiple of 8. If the data to be authenticated is in multiples of 8, then an 80h byte is appended followed by 7 00h for the last block.

The *IV_Seq* is the initialization vector of the RMAC computation. It is the concatenation of the 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 00h and a 2 byte SM-MAC sequence number. The SM-MAC sequence number starts at 00 00h from the last GET CHALLENGE and incremented every time a SM-MAC command is executed. Calling GET CHALLENGE again will 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.

6.5.2. 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 04h. 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 00h
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 **SM for ISO-in Command**.

6.5.3. 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.

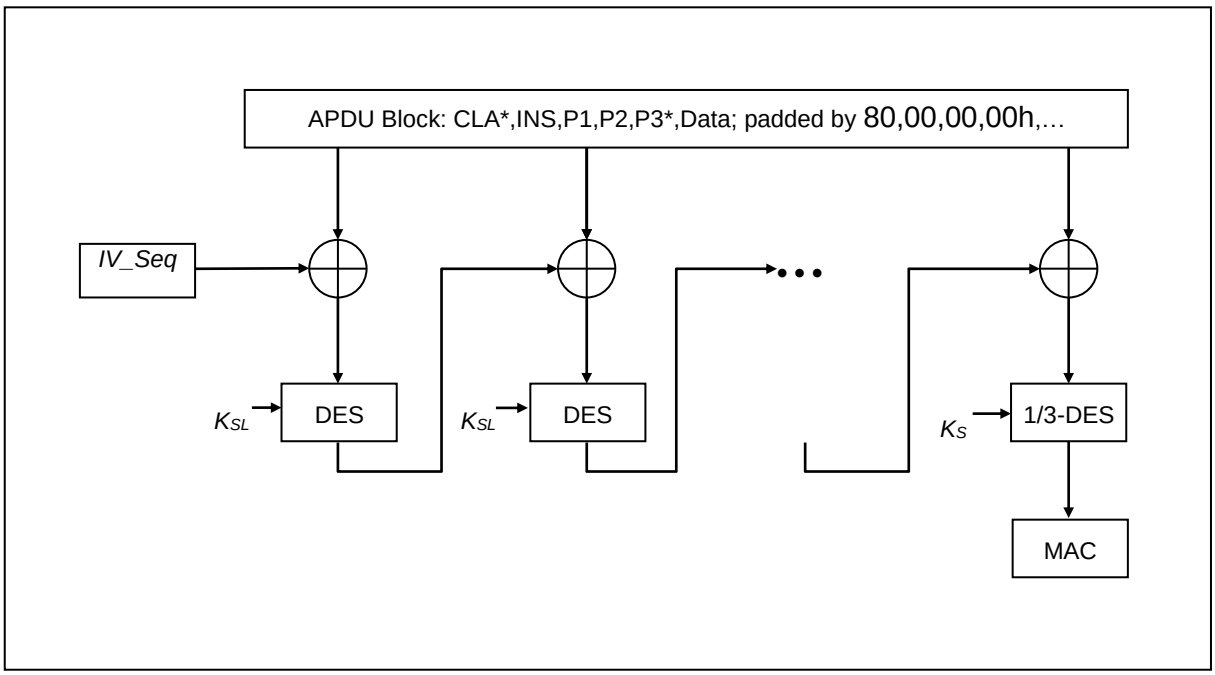


Figure 8: Secure Messaging for Authenticity Retail-MAC

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 00h 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 80h followed by zeroes. If the APDU block is a multiple of 8, the last block will then be: 80, 00, 00, 00, 00, 00, 00, 00h.
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 **SM for ISO-in Command** and **SM for ISO-out Command** respectively.
5. After every computation of SM-MAC, the sequence number in IV_Seq is incremented.

6.6. Secure Messaging for Confidentiality (SM-ENC)

ACOS6-SAM Version 4.02 and above support 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 on the appropriate security conditions byte; see **Security Attribute Compact (SAC)** for more details. Almost all the commands can also use secure messaging initiated by the terminal. The commands that do not accept secure messaging are GET CHALLENGE, MUTUAL AUTHENTICATION and GET RESPONSE.

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

Sections from **Notations** to **Secure Messaging Data Object** discuss the constructs used in secure messaging. **Secure Messaging Semantics** details the exact constructs of secure messaging commands and response. **SM Specific Response Status Bytes** lists the secure messaging return codes.

6.6.1. Notations

To describe the two SM modes adequately, there are some notations to be introduced in this section. The following are the possible ISO 7816 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 **Secure Messaging Semantics** has the SM bits b3 and b2 set to 1. That is, the original CLA XOR 0Ch.
- 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:

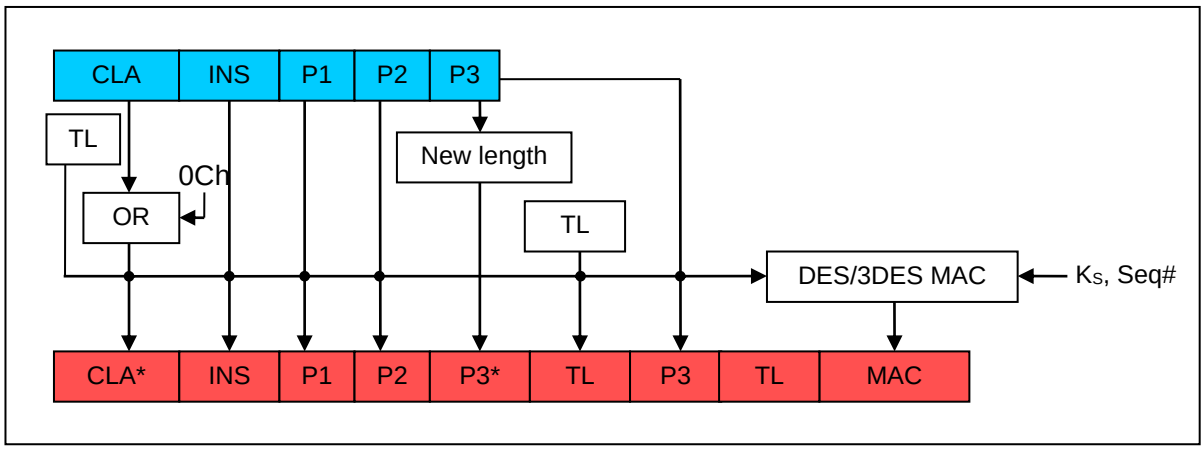


Figure 9: Secure Messaging for Confidentiality ISO-OUT Command Transformation

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

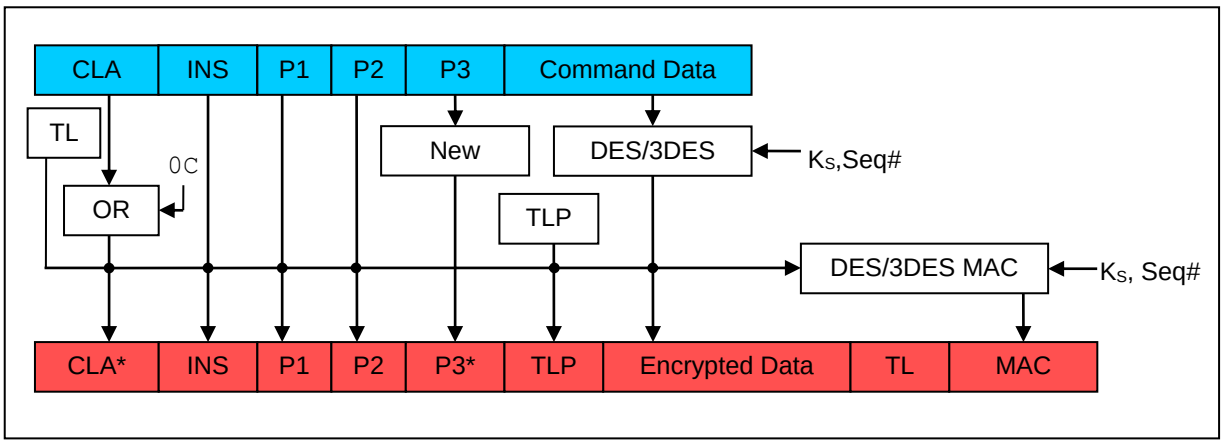


Figure 10: Secure Messaging for Confidentiality ISO-IN Command Transformation

The following figure shows the response transformation:

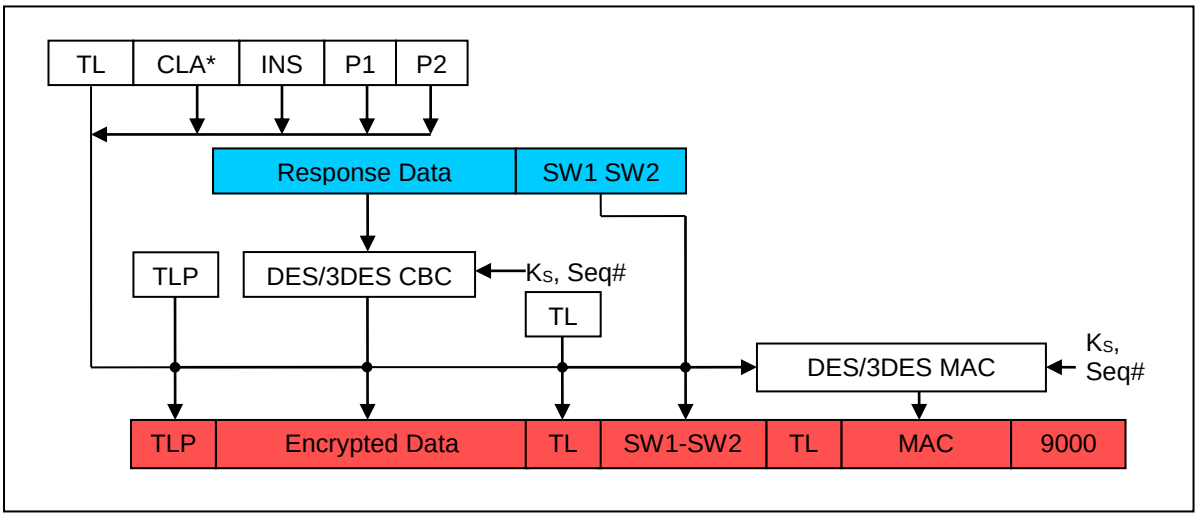


Figure 11: Secure Messaging for Confidentiality Response Transformation

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, and are described in **Secure Messaging Data Object**.

6.6.2. Secure Messaging Key

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

6.6.3. 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 FFh, the next number will be 00 00 00 00 00 00 00 00h.

6.6.4. 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 **Computing the Secure Messaging Retail MAC (RMAC)**. 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 = 69 88h), 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 re-established.

6.6.5. Confidentiality

The COS and terminal will use the SM key and DES/DES 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.

6.6.6. 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 multiples of 8, an 80h byte is appended followed by 0 to 6 00h 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.

6.6.7. 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-SAM SM-ENC:

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

Table 20: 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.

6.6.8. Secure Messaging Semantics

Card commands that are basically input and output commands are 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 **Notations** will be modified with SMDOs. The following two sections show how those modifications are to be done.

6.6.8.1. ISO-IN

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

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

CLA* = CLA OR 0Ch
P3* = 3 + n* + 2 + 4

L_{87}* = n* + 1 (Length of Tag 87h)
Pi = Padding indicator: 00h – No padding is used in encrypted data
01-07h – Number of padding bytes used in encrypted data
Encrypted Data* = ENC_CBC (<Command Data> padding, ++Seq#)
n* = P3 + length (padding) = P3 + Pi
MACcmd* = SIGN_CBC (<89 04h CLA* INS P1 P2> <87h 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 61 0Ch. Note that 61 0Ch 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 = 0Ch exactly. Otherwise, 6C 0Ch will be replied without SM.

The response to GET RESPONSE is as follows:

	99h	02h	SW1 SW2	8Eh	04h	MAC_{rsp}	90 00h
Bytes	1	1	2	1	1	4	2

MACrsp = SIGN_CBC (<89 04h CLA* INS P1 P2> <99 02h SW1 SW2> padding, ++Seq#)

The field SW1 SW2 is the actual status bytes returned for the command. The last status bytes 90 00h states that the GET RESPONSE is successful.

6.6.8.2. ISO-OUT

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

	CLA*	INS	P1	P2	P3*	97h	01h	P3	8Eh	04h	MAC_{cmd}
Bytes	1	1	1	1	1	1	1	1	1	1	4

CLA* = CLA OR 0Ch
P3* = 9
MACcmd = SIGN_CBC (<89 04h CLA* INS P1 P2> <97 01h P3> padding, ++Seq#)

If the command accepts secure messaging, the key has been established and the MAC_{cmd} is correct, the response will be 61 xx. Same as ISO-in, the status bytes of 61 xx 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.

Note that the get response must be called with P3 = xx exactly, else 6C xx will be returned. This is to prevent man in the middle attacks. The original data out *n* must be less than or equal to 240 bytes. Else, error status 67 00h will be returned.

The response is as follows:

	87h	L_{87}	Pi	Encrypted data	99h	02h	SW1 SW2	8Eh	04h	MAC_{rsp}	90 00h
Bytes	1	1	1	n*	1	1	2	1	1	4	2

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

***Note**: 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.*

6.6.8.3. ISO-IN-OUT

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

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

CLA* = CLA OR 0Ch
P3* = 3 + n* + 2 + 4
L_{87} = n* + 1 (Length of Tag 87h)
Pi = Padding indicator: 00h – No padding is used in encrypted data
01-07h – Number of padding bytes used in encrypted data
Encrypted Data = ENC_CBC (<Command Data> padding, ++Seq#)
n* = P3 + length (padding) = P3 + Pi
MACcmd = SIGN_CBC (<89 04h CLA* INS P1 P2> <87h 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 61 xx. Same as ISO-in, the status bytes of 61 xx 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.

Note that the get response must be called with P3 = xx exactly, else 6C xx will be returned. This is to prevent man in the middle attacks. The original data out *n* must be less than or equal to 240 bytes. Else, error status 67 00h will be returned.

The response is as follows:

	87h	L_{87}	Pi	Encrypted data	99	02h	SW1 SW2	8Eh	04h	MAC_{rsp}	90 00h
Bytes	1	1	1	n*	1	1	2	1	1	4	2

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

Note: *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.*

6.6.9. SM Specific Response Status Bytes

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

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

Table 21: Secure Messaging – Specific Return Codes

6.7. 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, and PIN File) to provide security for the files in question. **Figure 12** provides an example of how all these files work together.

In this example, the current DF has an SE file of F0 00h. 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 01h. In the DF's key file and PIN file the record of 01h 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 02h. Then, it will check if the PIN ID of 02h in the PIN file has been verified beforehand.

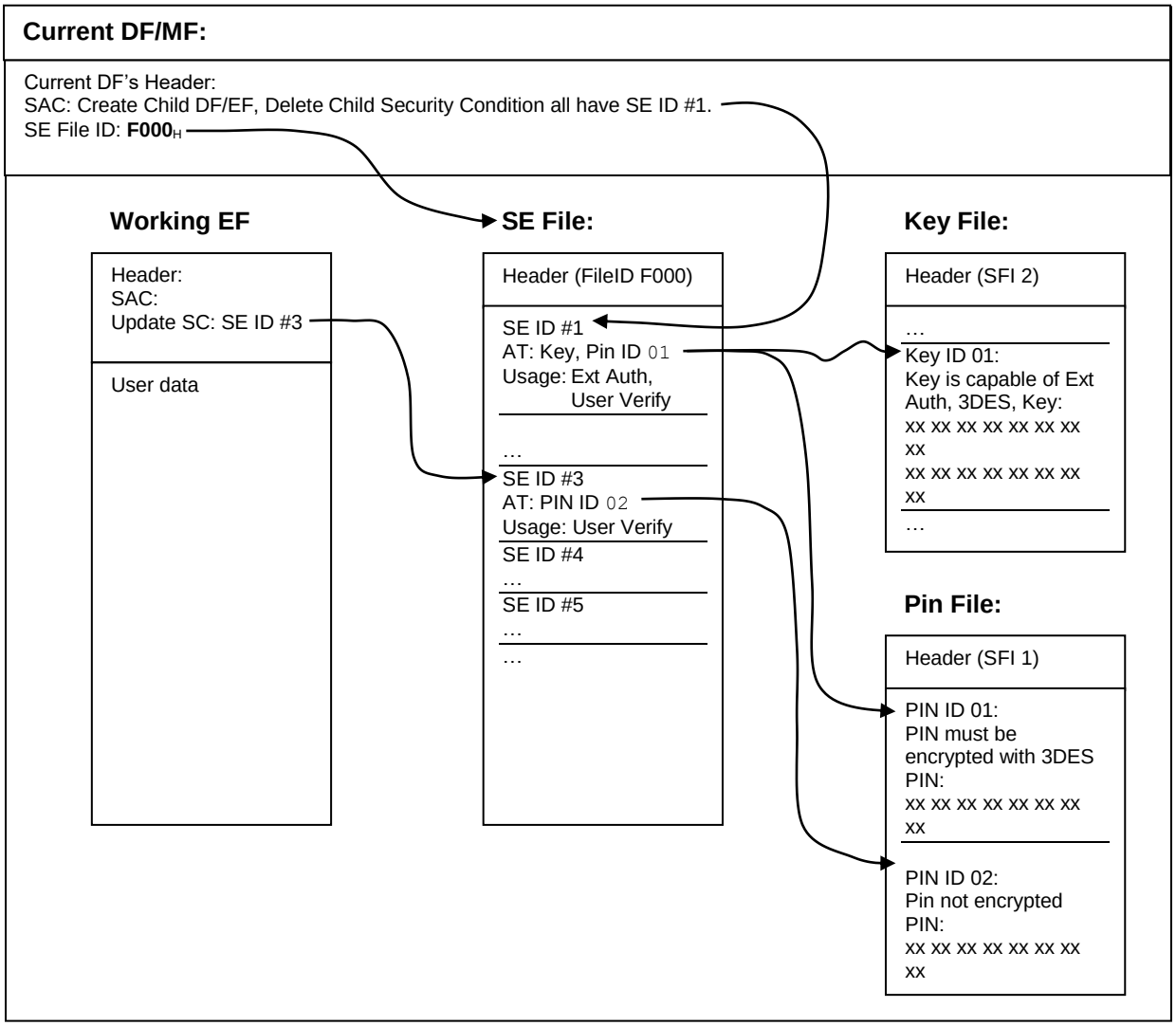


Figure 12: Relationship Between Working EFs and Internal Security Files

6.8. Encrypted Code Operations

Depending on the setting of the PIN records in EF1 (**PIN Data Structure**), 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.

6.8.1. 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 a 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 KS has been established:

Terminal		ACOS6-SAM
Compute R = ENC (C_i, K_S)		
Submit encrypted code 00 20 00h *i* 08h *R*	➔	Check if code identifier *i* and session key K_S is valid. Verify that R = ENC (C_i, K_S)
	←	Returns 90 00 if successful

Figure 13: Submit Encrypted Code Procedure

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

6.8.2. 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 **Submit Encrypted Code**.
Get new PIN Compute R = DEC (new PIN, K_S) Change PIN 00 24 00h *i* 08h *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 90 00h if successful

Figure 14: 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.

6.9. 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, ACOS7 or ACOS10 card. For the purpose of key injection, we shall refer to the ACOS6-SAM with the key to inject as the "*source SAM*" and the ACOS6/ACOS6-SAM/ACOS7/ACOS10 to receive the key as the "*target SAM*."

This function allows for a master and subordinate SAM relationships. 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 (number used once) initial vector*. The session key is generated by the mutual authentication as in **Mutual Authentication with ACOS**. 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 as 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 or later.*

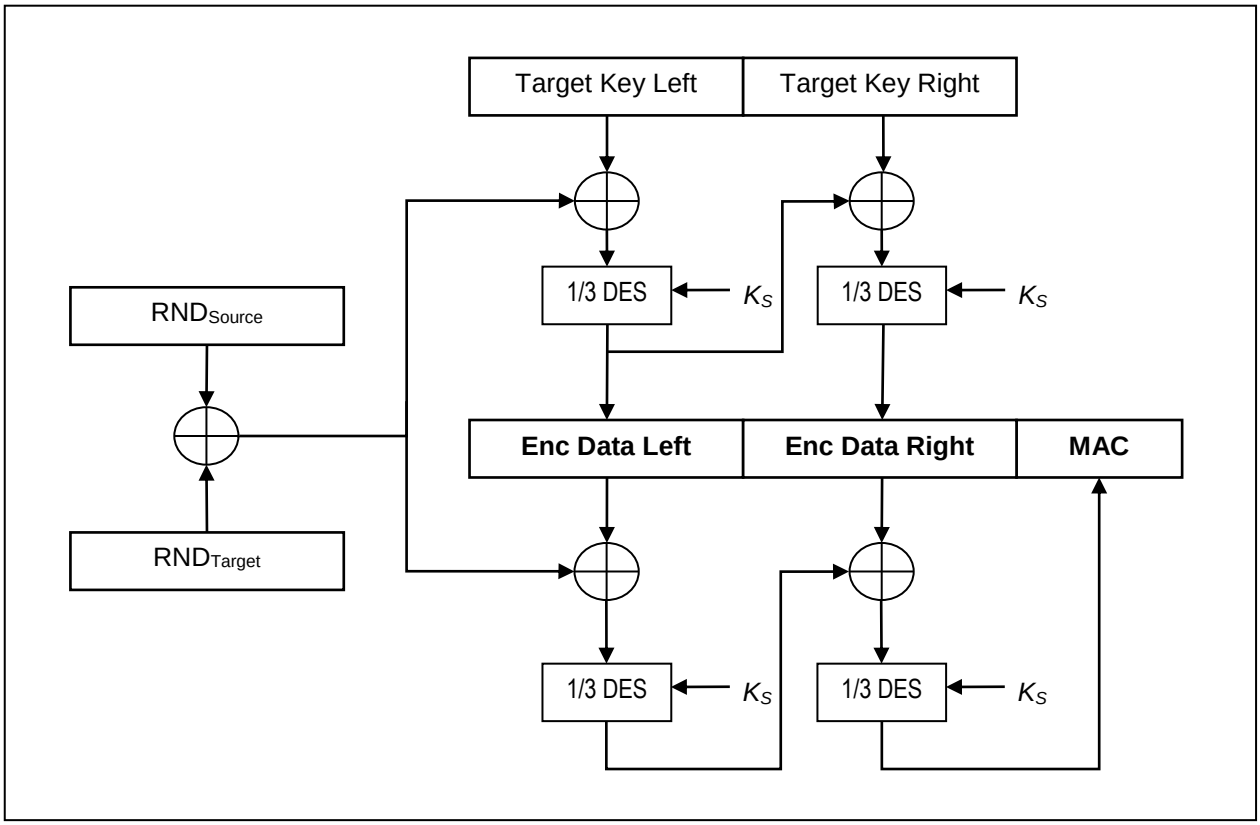


Figure 15: Key Injection Encryption Computation

6.10. Anti-tearing Mechanism

ACOS6-SAM uses an anti-tearing mechanism in order to protect the card from data corruption due to card tearing, which happens when the card is suddenly pulled out of the reader during data update, or when the reader suffers from 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.

7.0. Commands

This contains the command set of the card, excluding the SAM specific commands. Most of these commands are defined in ISO 7816 Part 4. Some instruction codes i.e., INS, are the same. These are distinguished by the class CLA byte.

Command	Instruction	Description
CREATE FILE	E0h	Create MF/DF/EF according to supplied parameters.
SELECT FILE	A4h	Select MF/DF/EF files.
READ BINARY	B0h	Read data from referenced binary file.
UPDATE BINARY	D6h	Update data from referenced binary file.
READ RECORD	B2h	Read record data from referenced record file.
UPDATE RECORD/WRITE RECORD	D2h/DCh	Update record data from referenced record file.
APPEND RECORD	E2h	Append a record into the linear variable record file.
ACTIVATE FILE/CARD	44h	Activate a file, which ensures all security attributes of the file. Activating the card sets card from personalization state to user state.
DEACTIVATE FILE/CARD	04h	Deactivate the referenced file to disable access. Deactivating the card sets card from user state back to personalization state.
TERMINATE DF	E6h	Set the target DF to TERMINATED state.
TERMINATE EF	E8h	Set the target EF to TERMINATED state.
DELETE FILE	E4h	Delete the last created file.
GET CHALLENGE	84h	Get card random challenge data for authentication purpose.
MUTUAL/EXTERNAL AUTHENTICATION	86h	Perform mutual authentication or short key external authentication.
VERIFY	20h	Verify a code.
CHANGE CODE	24h	Change a code.
GET CARD INFO	14h	Get information about card serial number.
GET RESPONSE	C0h	Get result of a previous command.
CLEAR CARD	30h	Clear all EEPROM data and return the card to the pre-perso state.
SET KEY	DAh	Inject key from ACOS6-SAM or terminal.
GENERATE KEY	88h	Generate a diversified key to load into a client card.

Command	Instruction	Description
DIVERSIFY (OR LOAD) KEY DATA	72h	Diversify or load a key internally for ciphering operations.
ENCRYPT	74h	Encrypt data using session, diversified or loaded keys.
DECRYPT	76h	Decrypt data using session, diversified or loaded keys.
PREPARE AUTHENTICATION	78h	Perform first half of ACOS authentication procedure.
VERIFY AUTHENTICATION	7Ah	Perform second half of ACOS authentication procedure.
VERIFY ACOS INQUIRE ACCOUNT	7Ch	Verify ACOS3/6 inquire account purse command.
PREPARE ACOS ACCOUNT TRANSACTION	7Eh	Create a debit/credit command for ACOS card.
VERIFY DEBIT CERTIFICATE	70h	Verify ACOS3/6's debit certificate.
GET KEY	CAh	Export encrypted key to inject into an ACOS6 client card or another ACOS6-SAM card.

Table 22: Command Table Summary

7.1. CREATE FILE

This command is used to create a MF/DF/EF based on a set of attributes.

APDU	Description
CLA	00h - Clear Mode 04h - SM Mode
INS	E0h
P1	00h
P2	00h
P3	Length of Data
Data	FCP TLV of File to be created

The FCP TLV has tag of 62h. 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
80h	02	Size (in bytes) of the Transparent EF			O						If not specified, default is 0000h
82h	01	File Descriptor Byte (FDB)	M	M	M	M	M	M	M	M	See **File Descriptor Byte (FDB)** for valid values
	02	FDB || DCB	O	O	O	O	O	O	O	O	If DCB is not specified, default is 00h
	05	FDB || DCB || 00 || MRL || NOR				O	O	O	O	O	If MRL/NOR is not specified, default is 00h
	06	FDB || DCB || 00 || MRL || 00 || NOR				O	O	O	O	O	If MRL/NOR is not specified, default is 00h
83h	02	File ID	M	M	M	M	M	M	M	M	See **File ID** for valid values
84h	<= 10h	DF Long Name	O	O							
88h	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.
8Ah	01	Life Cycle State Integer	O	O	O	O	O	O	O	O	If not specified, the default is 01h.
8Ch	<= 08	Security Attributes Compact	O	O	O	O	O	O	O	O	See **Security Attribute Compact (SAC)** for more details
ABh	<= 20h	Security Attributes Extended	O	O							See **Security Attribute Expanded (SAE)** for more details
8Dh	02	SE File ID	O	O							See **SE File ID (for DF only)** for more details
87h	02	FCI File ID	O	O							See **FCI File ID (for DF only)** 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: 3Fh (MF), 01h (Transparent EF), 02h (Linear Fixed EF), 04h (Linear Variable EF), 06h (Cyclic EF), 0Ch (Internal LV EF or KEY EF), and 0Eh (Internal Cyclic EF, or Purse EF).

File IDs cannot be: FFFFh, 0000h and 3FFFh.

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

The MF's file ID should always be 3F00h, and should always be the first 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 IDs/DF Names under a DF.

Refer to **SAM Scenarios for MIFARE Products** and **Sample Card Initialization** for examples on using this command.

Specific Response Status Bytes

SW1 SW2	Description
6A 86h	Incorrect P1/P2
67 00h	Wrong P3, must be consistent with the Length of the FCP TLV
6A 80h	Wrong FCP Tag - should be 62h
	FCP contains invalid TLVs, unrecognized tags, or wrong lengths in TLVs
	Creating MF, but MF already exists
	File ID is 3F00h but FDB is not MF or MF already exists
	Invalid File ID, FDB, SFI, LCSI, etc.
62 83h	Current DF is terminated/ deactivated
69 82h	Security condition not satisfied
6A 89h	File ID/DF Name already exists
6A 84h	Not enough free space in card to create file

7.2. SELECT FILE

This command is used to select a target MF/DF/EF based on File ID or DF Name.

APDU	Description
CLA	00h
INS	A4h
P1	04h - if Data contains DF name and in CLEAR mode, else 00h
P2	00h
P3	00h - select MF, 02h - Data contains File ID, 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 = 61 XX. 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 **CREATE FILE**.

Specific Response Status Bytes

SW1 SW2	Description
62 83h	Target file is blocked, but is selected
69 82h	Target file has wrong checksum in header
69 86h	No MF found in card
6A 82h	File not found
6A 86h	Invalid P1/P2
67 00h	Wrong P3, P3 not compatible to P1/P2
61 nn	File selected successfully under ACOS6-SAM mode. Issue GET RESPONSE with P3 =NN in order to retrieve the file's FCI template

7.3. READ BINARY

This command is used to read out the contents of a Transparent File given the file offset.

APDU	Description
CLA	00h - Clear mode 04h - SM mode
INS	B0h
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 (See **Card Header Block** for valid EEPROM physical addresses).

If MF exists, this command follows the READ BINARY command specified in ISO 7816 Part 4.

Specific Response Status Bytes

SW1 SW2	Description
62 82h	Invalid offset
62 83h	Current DF is blocked or target EF is blocked
67 00h	Incorrect P3, length exceeds file size limit
69 81h	Wrong file type; target file must be TRANSPARENT EF
69 82h	Target file's header block has wrong checksum, or security condition not satisfied
69 86h	No EF selected
6A 82h	SFI not found
6B00h	Invalid P1/P2, invalid SFI
6C nn	Wrong P3; NN = maximum bytes available in file to read
6F 00h	Invalid physical address (when directly accessing EEPROM)

7.4. UPDATE BINARY

This command is used to write data to a Transparent File, given the offset.

APDU	Description
CLA	00h - Clear mode 04h - SM mode
INS	D6h
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 (See **Card Life Cycle States** for valid EEPROM physical addresses).

If MF exists, this command follows the UPDATE BINARY command specified in ISO 7816 part 4.

Specific Response Status Bytes

SW1 SW2	Description
62 82h	Invalid offset
62 83h	Current DF is blocked, or target EF is blocked
67 00h	Incorrect P3, length exceeds file size limit
69 81h	Wrong file type; target file must be TRANSPARENT EF
69 82h	Target file's header block has wrong checksum, or security condition not satisfied
69 86h	No EF selected
6A 82h	SFI not found
6B 00h	Invalid P1/P2, or invalid SFI
6C nn	Wrong P3; NN = maximum bytes available in file to update
6F 00h	Invalid physical address (when directly accessing the EEPROM), or write failure

7.5. READ RECORD

This command is used to read out the contents of a record block from a Record-based EF.

APDU	Description
CLA	00h - Clear mode; 04h - SM mode;
INS	B2h
P1	Record number based on P2
P2	If 5 MSb is not 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 – 000b

If the file's MRL or NOR field is zero, the COS will return 6A 83h 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
62 83h	Current DF is blocked, or Target EF is blocked
67 00h	Incorrect P3
69 81h	Wrong file type; target file must be RECORD-BASED EF
69 82h	Target file's header block has wrong checksum, or security condition not satisfied
69 86h	No DF selected, or no EF selected
6A 82h	SFI not found
6A 83h	Record not found, invalid record reference
6A 86h 6B 00h	Invalid P1/P2, invalid SFI
6C nn	Wrong P3, NN = maximum bytes available in file to read

7.6. UPDATE RECORD

This command is used to write the contents of a record block in a Record-based EF.

APDU	Description
CLA	00h - Clear mode 04h - SM mode
INS	DCh
P1	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
62 83h	Current DF is blocked, or Target EF is blocked
67 00h	Incorrect P3
69 81h	Wrong file type; target file must be RECORD-Based EF
69 82h	Target file's header block has wrong checksum, security condition not satisfied
69 86h	No DF selected, no EF selected
6A 82h	SFI not found
6A 83h	Record not found, invalid record reference
6A 86h 6B 00h	Invalid P1/P2, invalid SFI
6C nn	Wrong P3, NN = maximum bytes to write to record

7.7. WRITE RECORD

This command does exactly the same thing as UPDATE RECORD.

APDU	Description
CLA	00h - Clear mode 04h - SM mode
INS	D2h
P1	Record number (in ACOS6-SAM mode based on P2)
P2	If 5 MSb <> 00000, it is SFI; 3 LSb
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
62 83h	Current DF is blocked, or Target EF is blocked
67 00h	Incorrect P3
69 81h	Wrong file type; target file must be RECORD-based EF
69 82h	Target file's header block has wrong checksum, security condition not satisfied
69 86h	No DF selected, no EF selected
6A 82h	SFI not found
6A83h	Record not found, invalid record reference
6A 86h 6B 00h	Invalid P1/P2 in ACOS3 mode Invalid P1/P2, invalid SFI in ACOS6-SAM mode
6C nn	Wrong P3, NN = maximum bytes to write to record

7.8. APPEND RECORD

This command is used to add a record at the end of a Linear Variable EF.

APDU	Description
CLA	00h - Clear Mode 04h - SM Mode
INS	E2h
P1	00h
P2	00h
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 first record whose first byte is FFh (a record whose first byte is FFh is considered 'empty', and therefore is at the 'end' of the file). P3 can be less than the file's MRL, in such case; FFh will be padded to the new record.

Specific Response Status Bytes

SW1 SW2	Description
62 83h	Target file/current DF is blocked
69 81h	Target file is not Linear Variable EF
69 82h	Target file has wrong checksum in header
69 82h	Security condition not satisfied
6A 83h	Record not found
69 86h	No file selected
6A 84h	No more empty record in EF
6B 00h	Wrong P1/P2
67 00h	Invalid P3

7.9. ACTIVATE FILE/CARD

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

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

If P1 = 00h, 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 = 01h, this command activates the card by setting the card life cycle fuse to 00h even when the card is in pre-personalization state.

Specific Response Status Bytes

SW1 SW2	Description
64 00h	Target file is terminated
69 82h	Target file has wrong checksum, or security condition not satisfied
6A 82h	File referenced not found
69 86h	No file selected
6A 86h	Invalid P1/P2
67 00h	Invalid P3, must be 2
6F 00h	EEPROM failure

7.10. DEACTIVATE FILE/CARD

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

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

If P1 = 00h, 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 = 01h, this command resets the card life cycle fuse to FFh. This command can only be called in MF level and bit 5 - Deactivate Card Enable Flag of the **Card Header Block** 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
64 00h	Target file is terminated
69 82h	Target file has wrong checksum, or security condition not satisfied
6A 82h	File referenced not found
69 86h	No file selected
6A 86h	Invalid P1/P2
67 00h	Invalid P3, must be 2
6F 00h	EEPROM failure

7.11. TERMINATE DF

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

APDU	Description
CLA	00h - Clear Mode 04h - SM Mode
INS	E6h
P1	00h
P2	00h
P3	02h – Terminate the DF File whose ID is referenced by the DATA 00h – 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
64 00h	Target file is terminated
69 82h	Target file has wrong checksum, security condition not satisfied
6A 82h	File referenced not found
69 86h	No file selected
6A 86h	Invalid P1/P2
67 00h	Invalid P3, must be 2
6F 00h	Update failure
69 81h	File is not DF type

7.12. TERMINATE EF

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

APDU	Description
CLA	00h - Clear Mode 04h - SM Mode
INS	E8h
P1	00h
P2	00h
P3	02h – Terminate the EF File whose ID is referenced by the DATA 00h – 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
64 00h	Target file is terminated
69 82h	Target file has wrong checksum, or security condition not satisfied
6A 82h	File referenced not found
69 86h	No file selected
6A 86h	Invalid P1/P2
67 00h	Invalid P3, must be 2
6F 00h	Update failure
69 81h	File is not EF type

7.13. DELETE FILE

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

APDU	Description
CLA	00h – Clear Mode 04h - SM Mode
INS	E4h
P1	00h
P2	00h
P3	02h – Delete the File whose ID is referenced by the DATA 00h – 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
64 00h	Target file is terminated
69 82h	Target file has wrong checksum, or security condition not satisfied
6A 82h	File referenced not found
69 86h	No file selected
6A 86h	Invalid P1/P2
67 00h	Invalid P3, must be 2
6F 00h	EEPROM failure
69 81h	File is not EF type
6A 80h	Target DF file has children Target file is not the last file in memory

7.14. GET CHALLENGE

This command generates a 4/8-byte Challenge Data for authentication purposes.

APDU	Description
CLA	00h
INS	84h
P1	00h
P2	00h
P3	04h/08h

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
67 00h	Incorrect P3
6A 86h	Wrong P1/P2

7.15. MUTUAL/EXTERNAL AUTHENTICATION

This command authenticates the referenced key(s) of the currently selected DF. There are two types of authentication in ACOS6-SAM: 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.

APDU	Description
CLA	00h
INS	82h
P1	Key index of Internal Key (key type bit 2 must be set); if MSb=1, use local EF2 else use global EF2 P1= 00 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	10h or 04h
Data	If P3=10h, command data is xDES (RNDc, Kt) || RNDt If P3=04h and P1=00h, command data is the left 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 90 00h.

If P3=10h, on success, COS will return 61 08h. The result is: xDES (RNDt. Ks).

Triple DES will be used if both Keys (referenced by P1 and P2) have ALGO field of 00h. In this case, if either key's length is less than 16, FFh will be padded.

Single DES will be used if either Key (referenced by P1 and P2) has ALGO field not equal to 00h. 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).

See **Mutual Authentication** for a more detailed description of the process.

Specific Response Status Bytes

SW1 SW2	Description
6A 87h	KEY(s) referenced in P1/P2 not capable of Internal/External authentication
63 Cnh	Wrong Crypto Data, Operation Fail, n tries left for KEY
67 00h	Incorrect P3, must be 16
69 83h	Terminal KEY is locked or Card Key is locked
69 85h	GET CHALLENGE command not previously called
6A 86h	Invalid P1/P2
69 86h	No DF selected

SW1 SW2	Description
62 83h	Current DF is blocked, EF2 is blocked
6A 88h	EF2 not found
6A 83h	Key not found in EF2, key has invalid length
69 81h	Invalid EF2 (FDB, MRL, etc., not consistent)
61 08h	Authentication OK

7.16. VERIFY

This command submits a PIN code to gain access rights.

APDU	Description
CLA	00h
INS	20h
P1	00h
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.

Access rights achieved will not be validated when a new DF is selected.

Specific Response Status Bytes

SW1 SW2	Description
62 83h	Current DF is blocked; EF1 is blocked
63 Cnh	Verify fail, n tries remaining
67 00h	Incorrect P3, does not match PIN length, must be <= 32
69 81h	EF1 has wrong FDB, EF1 is not a valid PIN file
69 83h	Referenced PIN is locked
69 86h	No DF selected
6A 83h	PIN ID referenced in P1 is not found in EF1
6A 86h	Invalid P1/P2
6A 88h	EF1 not found
69 85h	Session Key not valid, or not consistent with key DES mode (Session key is needed for encrypted PIN)

7.17. CHANGE CODE

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

APDU	Description
CLA	00h
INS	24h
P1	00h
P2	If ACOS6-SAM mode: PIN ID: if MSb = 1, use Local PIN, else use Global PIN
P3	20h or less, or 08h if PIN is encrypted
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.

Specific Response Status Bytes

SW1 SW2	Description
62 83h	DF is blocked; EF1 is blocked
67 00h	Incorrect P3, does not match KEY length must be <= 32
69 81h	EF1 has wrong FDB, EF1 is not a valid PIN file
69 83h	Referenced PIN is locked
69 86h	No DF selected
6A 83h	Referenced PIN ID not found in EF1
6A 86h	Invalid P1/P2
6A 88h	EF1 not found
69 85h	Session Key not valid, not consistent with key DES mode
69 66h	PIN_ALT of the PIN is disabled
69 82h	PIN not verified previously

7.18. GET CARD INFO

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

APDU	Description
CLA	80h
INS	14h
P1	See options from the next table
P2	
P3	

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

P1	P2	P3	Description
00h	00h	08h	Returns card's unique serial number
01h	00h	00h	Returns the number of files under the currently selected DF in SW1 SW2 = 90 XX, where XX is the number of files.
02h	xx	08h	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};
04h	00h	06h	Card returns the 6-byte Card ID Number (See **Card Header Block**).
05h	00h	00h	Returns the size of EEPROM in the status bytes SW1 SW2 = 90 XX.
06h	00h	08h	Returns the version number of the ACOS6-SAM in the form of ACOS6 Revision XX YY ZZh (41 43 4F 53 06 XX YY ZZh)

Specific Response Status Bytes

SW1 SW2	Description
67 00h	Incorrect P3
6A 80h	Wrong P1/P2, data not available

7.19. GET RESPONSE

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

APDU	Description
CLA	00h
INS	C0h
P1	0h
P2	0h
P3	Bytes to receive

Specific Response Status Bytes

SW1 SW2	Description
6A 86h	Wrong P1/P2
6C nn	Incorrect P3, P3 must be nn
69 85h	No data available

7.20. CLEAR CARD

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

APDU	Description
CLA	80h
INS	30h
P1	00h
P2	00h
P3	00h

On success, the card's entire EEPROM memory will be erased (set to FFh). This command is only available if the card's header info field: Card Life Cycle Status (address EEC7h) is not set (FFh). 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 is cleared.

Specific Response Status Bytes

SW1 SW2	Description
6F 00h	Command not available

7.21. SET KEY

This command 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 are protected by encryption and message authentication codes.

If bit 7 - Key Injection Only Flag of the **Card Header Block** 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 **Mutual Authentication**.

APDU	Description
CLA	80h
INS	DAh
P1	00h
P2	Key ID
P3	1Ch
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. See **Key Injection** for more information.

Specific Response Status Bytes

SW1 SW2	Description
69 85h	GET CHALLENGE command not previously called or session key not ready
62 83h	Current DF is blocked, or Target EF is blocked
69 86h	No DF selected
69 81h	Wrong file type of Key file, it should be Internal Linear Variable File
69 82h	Target file's header block has wrong checksum, or security condition not satisfied
6A 86h	Invalid P1 or P2
67 00h	Incorrect P3
6A 83h	Target Key is not ready or Key Length less than 16
90 00h	Success

8.0. SAM Specific Commands

This section contains the SAM specific commands. For command flows of how to use these command, see **SAM Scenarios**.

8.1. GENERATE KEY

This command is used to generate a diversified key to load into the ACOS3/6 card or other cards from deviation data such as a client card serial number. This command is catered for client card issuance purposes.

APDU	Description	
CLA	80h	
INS	88h	
P1	00h	Generate 8 Byte Key
	01h	Generate 16 Byte Key
	02h	Generate 24 Byte Key
P2	Key index of Master Key to generate Derived Key	
P3	08h	
Data	Input Data	

The master key must be capable of performing Internal Authenticate in its key type attribute. 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 the 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 a 16-byte 3DES key, first issue this command to generate the first 8 bytes of the 16-byte 3DES key. For the second part, bit-wise complement the plain text of serial number and issue this command again.

Derived Key by DES Calculation		Key Length (bytes)	
		8	16
Key A	ENC ([Input Data], Master Key)	X	X
Key B	ENC ([Input Data] **XOR** (FFFFFFFFFFFFFFFFh), Master Key)		
Key C	ENC ([Input Data] **XOR** (FFFFFFFF00000000h), Master Key)		X
Derived Key by AES Calculation			
Key D	ENC ([Input Data] | [0000000000000000h], Master Key)		

Table 23: Derived Key Calculation

P1	Algorithm Ref. of Master Key	Response Data
00h	DES/3DES/3KDES	Key A
00h	AES	Left 8 byte of Key D
01h	DES/3DES/3KDES	Key A | Key B
01h	AES	Key D
02h	DES/3DES/3KDES	Key A | Key B | Key C
02h	AES	Key D | Left 8 byte of Key D

Table 24: Response Data of Generate Key Command

Specific Response Status Bytes

SW1 SW2	Description
69 86h	No DF selected
6A 86h	Invalid P1 or P2
67 00h	Incorrect P3, must be 08h
6A 83h	Referenced key record not found in EF2
69 81h	Invalid EF2 (record size, file type, etc.)
6A 88h	EF2 not found
62 83h	Current DF is blocked; EF2 is blocked
69 83h	Usage counter is zero.
69 82h	Security condition not satisfied
6A 87h	Referenced Master Key is not capable of 3-DES encryption
61 08h	Command completed, issue GET REPONSE to get the result

8.2. DIVERSIFY (OR LOAD) KEY DATA

This command prepares the SAM card to perform ciphering operations by diversifying and loading the key. It takes the serial number and CBC initial vector as command data input.

APDU	Description								
CLA	80h								
INS	72h								
P1	b7	b6	b5	b4	b3	b2	b1	b0	Description
	-	0	0	0	0	0	0	1	Secret Code (Sc)
	-	0	0	0	0	0	1	0	Account Key (K_{ACCT})
	-	0	0	0	0	0	1	1	Terminal Key
	-	0	0	0	0	1	0	0	Card Key
	-	0	0	0	0	1	0	1	Bulk Encryption Key (Not diversified)
	-	0	0	0	0	1	1	0	Initial vector
	0	-	-	-	-	-	-	-	16-byte Key
	1	-	-	-	-	-	-	-	24-byte Key
P2	Index of Master Key: Bit7: 1 = local Key in current EF2; 0 = global KEY EF2 Bit6-Bit5: 00b - RFU Bit4-Bit0: Key Index								
P3	If P1 = 1-4, P3 = 8/16,(if algo is AES, P3 = 8/16) If P1 = 5, P3 = 0 If P1 = 6, P3 = 8 (Algo of Master Key is DES/ 3DES/ 3KDES) P3 = 16 (Algo of Master Key is AES)								
Data	If P1 = 1-4 Client card's Serial Number, (if algo is AES, Data is Client card's Serial Number or Client card's Serial Number append with "0000000000000000") If P1 = 5, No command data. If P1 = 6, DES/3DES/3KDES/AES CBC initial vector.								

If P1 = 1 to 4, this command will generate a diversified Target Key by using the method in General Key command, please refer to **Table 23**.

The DERIVED KEY is 24-byte Length Key, if DES, 3DES or AES calculation is used, the left 8-byte or 16-byte will be used.

P1 bit 7	Algorithm Ref. of Master Key	Derived Key
0 – 16-byte Length	DES/ 3DES/ 3KDES	Key A | Key B | Key A
1 – 24-byte Length	DES/ 3DES/ 3KDES	Key A | Key B | Key C
0 – 16-byte Length	AES	Key D
1 – 24-byte Length	AES	Key D | Left 8 byte of Key D

Table 25: Derived Key

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

DERIVED KEY = Master Key

The Master Key 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 00h or it will be ignored.

If the length of Master Key is less than 24-byte, 00h will be padded at the end of the key

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 a 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
69 86h	No DF selected
6A 86h	Wrong P1, P1 must be 1 to 6
67 00h	Wrong P3, P3 must be 8 (or 0)
62 83h	Current DF is blocked, or EF2 is blocked
69 82h	Security condition not satisfied
6A 88h	EF2 not found
6A 83h	Referenced Master Key in EF2 not found
69 81h	Invalid EF2 (FDB, MRL, etc., not consistent)
6A 87h	Referenced KEY not capable of authentication
69 83h	Referenced Key is locked
90 00h	Target key generated, and ready in SAM memory

8.3. ENCRYPT

This command is used to encrypt data using DES or 3DES or AES with either:

1. The session key created by the mutual authentication procedure with an ACOS3/6, DESFire, DESFire EV1/EV2/Light or MIFARE Plus 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.

APDU	Description								
CLA	80h								
INS	74h								
P1	b7	b6	b5	b4	b3	b2	b1	b0	Description
	-	0	0	0	0	0	0	-	ECB Mode
	-	0	0	0	0	0	1	-	CBC Mode
	-	0	0	0	0	1	0	-	Retail MAC Mode
	-	0	0	0	0	1	1	-	MAC Mode
	-	0	0	0	1	0	0	-	Prepare ACOS3 SM command.
	-	1	0	0	1	0	1	-	MIFARE DESFire Encryption
	-	1	0	0	1	1	0	-	MIFARE DESFire EV1/EV2/Light Encryption
	-	0	0	0	1	1	1	-	CMAC
	-	0	1	0	0	0	0		MIFARE Plus Command
	-	0	1	0	0	0	1		MIFARE Plus Response
	0	-	-	-	-	-	-	0	3DES
	0	-	-	-	-	-	-	1	DES
	1	-	-	-	-	-	-	0	3K DES
	1	-	-	-	-	-	-	1	AES
	-	-	-	-	-	-	-	-	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 or b5=1, P2 must be 1 If P2 = 0h, P1 can be either 0 or 1								
P3	P3 < 128 If bit 3 of P1 not equal to 1 and bit 5 of P1 not equal to 1 - If P2 = 1-3, multiple of 8 (DES/3DES/3KDES) or 16 (AES) up to 128 bytes - If P2 = 0, 0								

APDU	Description
Data	Plain text If P1 b6 = 1, The DATA format should be: • Length of Plain text data • Length of Command and Header of DESFire Card • Command and Header of DESFire Card • Plain text If P1 = A1h, the encryption is for a MIFARE Plus command • if MFP Command is *value* operations command, the DATA format should be Command code(1 BYTE)+BlockNum(2/4 BYTE)+Value(4 BYTE). • if MFP Command is *Proximity Check*, the DATA format should be Command code(1 BYTE)+ PPS1(1 BYTE). • if MFP Command is *Read*, the DATA format should be Command code(1 BYTE)+ BlockNum(2 BYTE) • if MFP Command is *Write*, the DATA format should be Command code(1 BYTE)+ BlockNum(2 BYTE) +plaintext If P1=A3h, • The data return by ICC (don't include SC code and don't include RMAC if RMAC exist) If authenticate with Authenticate EV2 First/ Authenticate EV2NonFirst command before, then P1 should be CDh or 8Fh. • If P1=CDh, the encryption is for a DESFire EV2/Light command. The DATA format should be: Plain text data • If P1=8Fh, the CMAC is for a DESFire EV2/Light command or response. The DATA format should be: ➢ EV2 Command: The whole command without the CMAC data ➢ EV2 Response: The whole response from EV2 ➢ Light Command: The whole command without the CMAC data and the CLA/P1/P2/Lc/Le bytes. ➢ Light Response: The SW2 and Response data from Light. If the EV2/Light card return the status code and plain text data only, then the data should be 00h, which should be sent to the SAM card.

After DESFire Encryption, DESFire EV1/EV2/Light Encryption and CMAC Calculation, the IV block will be updated.

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.

For MIFARE Plus, all the commands (Read/Write/Value/ProximityCheck) and responses which are sent to MIFARE Plus or received from MIFARE Plus should be sent to SAM to keep track of the various counter states of the card.

For DESFire EV2 and P1=8Fh, If authenticate with Authenticate EV2 First/ Authenticate EV2NonFirst command before, the response data from SAM card should be sent to the EV2 card directly.

For DESFire Light and P1=8Fh, If authenticate with Authenticate EV2 First/ Authenticate EV2NonFirst command before, the last 8 bytes of the response data from the SAM card are the CMAC data.

See scenarios in **SAM Scenarios** and **SAM Scenarios for MIFARE Products** for more information.

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

This command will compute a DES/3DES/AES 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**. If encrypting data with *Ks*, mutual authentication must also be done.

Specific Response Status Bytes

SW1 SW2	Description
69 86h	No DF selected
6A 86h	Invalid P1 or P2
67 00h	Incorrect P3
6A 83h	ACOS Target Key is not ready (use Diversify to generate the key)
61 XX	Encryption is done, use GET RESPONSE to get the result

8.4. DECRYPT

This command is used to decrypt data using DES or 3DES or AES with either:

1. The session key created by the mutual authentication procedure with an ACOS3/6, MIFARE DESFire, MIFARE DESFire EV1/EV2/Light or MIFARE Plus 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.

Verify and Decrypt ACOS3 SM Response:

APDU	Description								
CLA	80h								
INS	76h								
P1	b7	b6	b5	b4	b3	b2	b1	b0	Description
	-	0	0	0	0	0	0	-	ECB Mode
	-	0	0	0	0	0	1	-	CBC Mode
	-	0	0	0	1	0	0	-	Verify and Decrypt ACOS3 SM Response
	-	1	0	0	1	0	1	-	MIFARE DESFire Decryption
	-	1	0	0	1	1	0	-	MIFARE DESFire EV1/EV2/Light Decryption
	-	0	1	0	0	1	0	-	MIFARE Plus Decryption
	0	-	-	-	-	-	-	0	3DES
	0	-	-	-	-	-	-	1	DES
	1	-	-	-	-	-	-	0	3K DES
	1	-	-	-	-	-	-	1	AES
	0	0	0	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	P3 < 128 If P1 = A5h, P3=16/32/48 If bit 3 of P1 not equal to 1 - If P2 = 1-3, multiple of 8 (DES/3DES/3KDES) or 16 (AES) up to 128 bytes - If P2 = 0, 0								

APDU	Description
Data	Ciphertext If P1 = A5h, The DATA is Encrypted text If P1 b6 = 1, The DATA format should be: • Length of Plain text data, if unknown, use 00 • Length of Command and Header of DESFire Card • Command and Header of DESFire Card • Encrypted text If authenticate with Authenticate EV2 First/ Authenticate EV2NonFirst command before, and the EV2/Light card return the encrypted and CMAC data, then P1 should be CDh. The DATA format should be: • EV2 Response: The whole response from EV2 • Light Response: The SW2 and Response data from Light.

After DESFire Decryption or DESFire EV1/EV2/Light Decryption, the IV block will be updated.

After MIFARE Plus Decryption, the IV block will not be updated.

For MIFARE Plus, the current decrypt function is to perform Decrypt on the encrypted text obtained from MIFARE Plus by Read command.

For DESFire EV2/Light, If authenticate with Authenticate EV2 First/ Authenticate EV2NonFirst command before, the response data from SAM card are the plain text data.

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 **Secure Messaging Computation for ACOS3 – Method 2** for more information.

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

This command will compute a DES/3DES/AES decryption using the derived key specified in P2, and the ciphertext command data. If DES mode is Single DES, the second half of the derived key will be ignored. The key loading must first be performed with Load Key (See **DIVERSIFY (OR LOAD) KEY DATA**). If decrypting data with Ks, mutual authentication must also be done.

Specific Response Status Bytes

SW1 SW2	Description
69 86h	No DF selected
6A 86h	Invalid P1 or P2
67 00h	Incorrect P3
6A 83h	ACOS Target Key is not ready (use Diversify to generate the key)
61 XX	Decryption is done, use GET RESPONSE to get the result

8.5. PREPARE AUTHENTICATION

This command is used to authenticate the SAM card (as the terminal) to the ACOS 3/6 card or MIFARE Ultralight C/MIFARE DESFire Card/MIFARE DESFire EV1/EV2/Light /MIFARE Plus card.

APDU	Description
CLA	80h
INS	78h
P1	00h – 3DES 01h – DES 02h – 3KDES (MIFARE DESFire EV1 /ACOS3) 03h – AES (MIFARE DESFire EV1/ EV2/Light /MIFARE Plus/ACOS3) 80h – 3DES (MIFARE DESFire Authenticate only) 81h – DES (MIFARE DESFire Authenticate only) Other – RFU
P2	0h – Verify ACOS3/6 Authenticate Return 01h – MIFARE Ultralight C/DESFire Authenticate by (Diversified) Terminal Key 05h – MIFARE Ultralight C/DESFire Authenticate by Bulk Encryption Key 02h – MIFARE Plus Authenticate. First Authenticate of SL1 to SL3 03h – MIFARE Plus Authenticate. Authentication in SL1 to SL2. 04h – MIFARE Plus Authenticate. Following Authenticate of SL2 to SL3. 06h – MIFARE DESfire EV2/Light AuthenticateEV2First /AuthenticateEV2NonFirst
P3	8 – (P1 = 00h, 01h, 02h, 80h, 81h) 16 – (P1 = 03h)
Data	Card Challenge Data

On success, the command will return SW1 SW2 = 61 nn. Issue GET RESPONSE with P3 = nn to get the Authenticate of ACOS card or MIFARE Ultralight C, MIFARE DESFire Card.

For ACOS3/6: If P2 = 00h: On success, the command will return SW1 SW2 = 61 10h. 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, see **Mutual Authentication with ACOS**.

For MIFARE Ultralight C or MIFARE DESFire: If P2 = 01h or 05h: On success, the command will return SW1 SW2 = 61 10h. Issue GET RESPONSE with P3 = 10h 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: *MIFARE Ultralight C authentication always uses triple DES. For more information, see* ***MIFARE Ultralight C****.*

For MIFARE Plus: If P2=02/03/04h, on success, the command will return SW1 SW2=61 20h. Issue GET RESPONSE with P3=20h 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. See **Mutual Authentication**.

Specific Response Status Bytes

SW1 SW2	Description
69 86h	No DF selected
6A 86h	Invalid P1 or P2
67 00h	Incorrect P3, must be 08h
6A 83h	ACOS Key (KT or KC) is not ready (use Diversify to generate this key)
69 82h	Security condition not satisfied
61 10h	Command completed, issue GET REPONSE to get the result

8.6. VERIFY AUTHENTICATION

This command is used to verify the ACOS 3/6, MIFARE Ultralight C, MIFARE DESFire/MIFARE DESFire EV1/EV2/Light or MIFARE Plus card to the terminal. The Session Key Ks will also be generated internally.

APDU	Description
CLA	80h
INS	7Ah
P1	00h – 3DES (P2 = 0) 01h – DES (P2 = 0) 02h – 3KDES (P2 = 0, ACOS3) 03h – AES (P2 = 0, ACOS3) Other – RFU
P2	00h – Verify ACOS3/6 Authenticate Return 01h – Verify MIFARE UL-C/ DESFire/ DESFire EV1 Authenticate Return 02h – Verify MIFARE Plus Authenticate return Or MIFARE Desfire EV2/Light AuthenticateEV2First/AuthenticateEV2NonFirst Return
P3	08h – (P2 = 0, P2 = 1 and Session Key is DES/3DES) 16h – (P2 = 1 and Session Key is 3KDES/AES) 16h – (P2=02, and MIFARE Plus return data ek(RndA')) 32h – (P2=02, and MIFARE Plus return data ek(TI+PICCcap2+PCDcap2))
Data	ACOS 3/6: DES (K_S, RND_T) MIFARE DESFire/ DESFire EV1/EV2/Light return data: ek(RndA') MIFARE Plus return data ek(RndA') or ek(TI+PICCcap2+PCDcap2)

If P2 = 02h, and the P2 of Prepare Authentication command is 03h, it will return 6 bytes Crypto1 session key. The terminal should use that XOR with the old Crypto1 key to generate a new Crypto1 key, and use the new MIFARE Crypto1 key to do Authentication(if in SL1, terminal may not use this session key). Otherwise, it will generate two AES session keys for internal calculating.

On success, the command will return SW1 SW2 = 90 00h. This means that the mutual authentication with the client card is successful, and the client card is authenticated.

Specific Response Status Bytes

SW1 SW2	Description
69 86h	No DF selected
6A 86h	Invalid P1 or P2
67 00h	Incorrect P3, must be 08h
6A 83h	ACOS-SAM Session Key or RND_T are not ready. Use PREPARE AUTHENTICATION to build these keys.
69 82h	Data is incorrect
90 00h	Data is correct, ACOS Mutual Authentication is successful

8.7. VERIFY ACOS INQUIRE ACCOUNT

This command is used to verify the ACOS3/6 card's Inquire Account purse command. It will verify that the MAC checksum returned by ACOS3/6 are correct with the SAM's diversified key. See **Secure Messaging Computation for ACOS3**.

APDU	Description								
CLA	80h								
INS	7Ch								
P1	b7	b6	b5	b4	b3	b2	b1	b0	Description
	-	0	0	0	0	-	0	-	ACOS INQ_AUT is disabled
	-	0	0	0	0	-	1	-	ACOS INQ_AUT is enabled
	-	0	0	0	0	0	-	-	ACOS INQ_ACC_MAC is disabled
	-	0	0	0	0	1	-	-	ACOS INQ_ACC_MAC is enabled
	0	-	-	-	-	-	-	0	3DES
	0	-	-	-	-	-	-	1	DES
	1	-	-	-	-	-	-	0	3K DES (ACOS3 only)
	1	-	-	-	-	-	-	1	AES (ACOS3 only)
P2	0h								
P3	1Dh								
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 first four 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 = 90 00h, meaning the client card's PURSE information is authentic and un-tampered.

Specific Response Status Bytes

SW1 SW2	Description
69 86h	No DF selected
6A 86h	Invalid P1 or P2
67 00h	Incorrect P3

SW1 SW2	Description
6A 83h	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.
6F 00h	Data Block’s MAC is incorrect
90 00h	Data Block’s MAC is correct

8.8. PREPARE ACOS ACCOUNT TRANSACTION

To create an ACOS3/6 Credit/Debit command, the MAC must be computed for ACOS3/6 to verify. See **Debit** and **Credit** for more details.

APDU	Description								
CLA	80h								
INS	7Eh								
P1	b7	b6	b5	b4	b3	b2	b1	b0	Description
	-	0	0	0	0	0	0	-	ACOS TRNS_AUT is disabled
	-	0	0	0	0	0	1	-	ACOS TRNS_AUT is enabled
	0	-	-	-	-	-	-	0	3DES
	0	-	-	-	-	-	-	1	DES
	1	-	-	-	-	-	-	0	3K DES (ACOS3 only)
	1	-	-	-	-	-	-	1	AES (ACOS3 only)
P2	E2h: Credit E6h: Debit								
P3	0Dh								
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 = E2h, use $TTREF_C$; if P2 = E6h, use $TTREF_D$

ATREF - this is the ATREF field returned by the INQUIRE ACCOUNT command

On success, SW1 SW2 = 61 0Bh, 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
69 86h	No DF selected
6A 86h	Invalid P1 or P2
67 00h	Incorrect P3, must be 0Dh
6A 83h	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.
61 0Bh	Command completed, issue GET RESPONSE to get the result

8.9. 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.

APDU	Description								
CLA	80h								
INS	70h								
P1	b7	b6	b5	b4	b3	b2	b1	b0	Description
	-	0	0	0	0	0	0	-	ACOS TRNS_AUT is disabled
	-	0	0	0	0	0	1	-	ACOS TRNS_AUT is enabled
	0	-	-	-	-	-	-	0	3DES
	0	-	-	-	-	-	-	1	DES
	1	-	-	-	-	-	-	0	3K DES (ACOS3 only)
	1	-	-	-	-	-	-	1	AES (ACOS3 only)
P2	0h								
P3	14h								
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
69 86h	No DF selected
6A 86h	Invalid P1 or P2
67 00h	Incorrect P3, must be 14h
6A 83h	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.
69 82h	Security condition not satisfied
6F 00h	DEBIT CERTIFICATE is invalid
90 00h	Success, DEBIT CERTIFICATE is valid

8.10. GET KEY

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

Get key command also allows secure key injection from the current SAM's Key File (SFI=02h) into ACOS7/10, MIFARE DESFire, MIFARE DESFire EV1/EV2/Light or MIFARE Plus Card with key diversification. Using this ensures the key 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** 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 **Mutual Authentication** (for ACOS6) or the MIFARE Plus/MIFARE DESFire mutual authentication procedure.

Note: *GET KEY command can only get the Key data.*

APDU	Description			
CLA	80h			
INS	CAh			
P1	Get Key for ACOS card Set Key			
	00h	Response data is Key in MSAM		
	01h	Response data is 16-byte Diversify Key		
	02h	Response data is 24-byte Diversify Key		
	03h	Response data is the Change Key command of MIFARE Plus Card		
	Get Key for DESFire card Change Key, Response data for DESFire/DESFire EV1 Change Key			
		Card Type	Authenticate Key No. And Changing Key No.*	Key Length
	80h	MIFARE DESFire	Are DIFFERENT in MIFARE DESFire card	16 bytes
	81h	MIFARE DESFire EV1	Are DIFFERENT in MIFARE DESFire EV1 card	16 bytes
	82h	MIFARE DESFire EV1	Are DIFFERENT in MIFARE DESFire EV1 card	24 bytes
	88h	MIFARE DESFire	Are the SAME in MIFARE DESFire card	16 bytes
	89h	MIFARE DESFire EV1	Are the SAME in MIFARE DESFire EV1 card	16 bytes
	8Ah	MIFARE DESFire EV1	Are the SAME in MIFARE DESFire EV1 card	24 bytes
P2	Key ID in SAM (New key for change)			
P3	If P1 = 00h, P3 is 08h If P1 = 01/02h, P3 is 10h If P1 = 03h, P3 is 0Bh If P1 = 80/81/82/88/89/8Ah: P3 is 0Bh or 0Dh			

APDU	Description
Data	If P1 = 00h, command data is RND_{Target} If P1 = 01/02h, command data is RND_{Target} + serial (or batch) number of target card If P1 = 03h - Serial Number for target card (8 Byte) - Write Command (A0 or A1) (1 Byte) - BNr (2 Byte) If P1 = 80/81/82/88/89/8Ah: - Serial Number for target card (8 Byte) - Original Key ID (Key in SAM card stored the Original key, 00 = Default Key of DESFire - Card) - Key No. (DESFire Card Key No.) - Key Version (DESFire Card Key Version, If not used, value = 00) - Command:C6 (Optional) - KeySetNo (Optional)

* Changing a *different* key means that the key to change is different from the authenticate key. Changing the *same* key means that the key to change is the same as the authenticate key in DESFire Card

If P1 = 00/01/02h, the response Data is the Set Key INPUT DATA of ACOS client card.

If P1 = 03h, the response Data is the Change Key command of MIFARE Plus Card.

If P1 = 80/88h, the response Data is the Change Key command of MIFARE DESFire Card.

If P1 = 81/89/82/8Ah, the response Data is the Change Key command of MIFARE DESFire EV1 Card

Please Set P1=81h/89h and P3=0Dh when changing the Desfire EV2 Key with the ChangeKeyEV2 command.

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

For injecting key into an ACOS6/ACOS6-SAM target card, a successful execution of this command will yield 611Ch. A Get Response will yield the encrypted key as:

RND_{Source} + Encrypted Key Data + MAC

The generation of Encrypted Key Data and MAC follows **Key Injection**. For more information, see the scenario in **Key Injection**.

For injecting key into MIFARE-based products please see the corresponding Change Key sections of **SAM Scenarios for MIFARE Products**.

Specific Response Status Bytes

SW1 SW2	Description
69 85h	SAM Session Key not ready
62 83h	Current DF is blocked, or Target EF is blocked
69 86h	No DF selected
69 81h	Wrong file type of Key file, it should be Internal Linear Variable File
69 82h	Target file's header block has wrong checksum, or security condition not satisfied
6A 86h	Invalid P1 or P2

SW1 SW2	Description
67 00h	Incorrect P3
6A 83h	Target Key is not ready or Key Length less than 16
61 1Ch	Success, use GET RESPONSE to get the result

9.0. Response Status Bytes

This section lists all the Card Response Status bytes (SW1 SW2) used in ACOS6-SAM, showing their general meaning. This section is meant for quick reference only. For meanings specific to a command, please see the corresponding command section.

SW1 SW2	Meaning
90 00h	Command executed successfully
91 XX	User file selected successfully
61 XX	XX encodes the number of data bytes available. Issue GET RESPONSE with P3 = XX to retrieve data
62 82h	Invalid offset
62 83h	Selected file terminated/deactivated
63 CX	Wrong authentication/verification data, X tries left for authentication key/PIN.
64 00h	Target file is terminated
67 00h	Wrong P3, P3 not compatible with P1/P2
69 66h	PIN_ALT of PIN is disabled
69 81h	Command incompatible with file structure
69 82h	Security status not satisfied or file header checksum incorrect
69 83h	Referenced PIN is locked
69 85h	Conditions of use not satisfied/no data available/MAC computation error in secure messaging (random number or session key not ready)
69 86h	Command not allowed (no currently selected DF/EF)/no MF on card
69 88h	Wrong MAC is submitted in secure messaging
6A 80h	Incorrect parameters in the command data field
6A 82h	File or application not found/card is in user terminated state
6A 83h	Record/Key not found
6A 84h	Not enough memory or record space
6A 86h	Incorrect parameters P1/P2
6A 87h	Key referenced cannot perform required authentication
6A 88h	Reference data/file not found
6A 89h	File already exists
6B 00h	Wrong parameters P1/P2
6C XX	Wrong P3, XX encodes the exact number of available data bytes
6D 00h	Invalid INS
6E 00h	Invalid CLA
6F 00h	Invalid physical address, EEPROM error, command not available.

Table 26: Response Status Bytes

10.0.SAM Scenarios

This section outlines the typical scenarios of using ACOS6-SAM cards along with an ACOS3/6 client card. The ACOS6-SAM should already be loaded with a file system and keys similar to that of **SAM Scenarios for MIFARE Products** or **Sample Card Initialization**. This section will demonstrate in APDU form how to perform several SAM and client card interactions. Notice that many scenarios that are demonstrated with ACOS3 client card also work with ACOS6 client card, with some modifications to the command set.

10.1. Personalizing ACOS3 KEYS/Codes

You can use the ACOS6-SAM to set the diversified keys or secret codes of ACOS3. 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, follow **GENERATE KEY** to generate the first and second half of the 16-byte 3DES key by multiple issuance of this command.

For ACOS6 client card, a similar procedure can be followed. Alternatively, the secure key injection procedure of **Key Injection** can be followed.

SAM	Terminal	ACOS3
	< Select Issuer DF < 00 A4 00 00 02h < XX XX < Submit Issuer PIN < 00 20 00 01 08h < XX XX…XX	
	Get Card Serial Number > (ACOS3) 80 14 00 00 08h >	< Serial Number
Check if Issuer PIN is submitted; Check if referenced master key is valid; Compute Key 61 08h >	< Generate Key$_J$ < 80 88 00 K_{MI} 08h < Serial Number	
Generated Key >	< Get Response < 00 C0 00 00 08h	
	Submit IC Code: 80 20 07 00 08h>< 8-byte IC Code >	< 90 00h
	Select FF 03h 80 A4 00 00 02 FF 03h >	< 90 00h
	Set Key$_J$ > Write key record with Key$_J$ > 80 D2 <Key$_J$ record> 00 08h > Generated Key >	< 90 00h

Figure 16: Personalizing ACOS3 Keys

10.2. Mutual Authentication with ACOS

The terminal will now use the SAM to compute the ACOS3/6/7/10 Session Key, and perform Mutual Authentication with the ACOS3 card. The SAM should know the Terminal Key and Card Key of the given ACOS3 since it is assumed that the keys are generated by the SAM.

SAM	Terminal	ACOS
	Get Card Serial Number >	
	(ACOS3) 80 14 00 00 08h >	
		< Serial Number
	< Generate Card Key using Master Key_I	
	< Diversify (@K_{MI}, Serial Number)	
Generate Card Key using Serial Number	< 80 72 04 # K_{MI}, 08h < Serial Number	
90 00h >		
	< Generate Terminal Key using Master Key_J	
	< Diversify (@K_{MJ}, Serial Number)	
Generate Terminal Key using Serial Number	< 80 72 03h #K_{MJ} 08h < Serial Number	
90 00h >		
	Get Challenge >	
	80 84 00 00 08h >	
		< RND_C
	< Prepare Authentication	
	< 80 78 00 00 08h < RND_C	
1. Generate RND_T		
2. Compute R = 3DES (RND_C, K_T)		
3. Compute Session Key K_S		
4. Form Response Block = R || RND_T		
61 10h >		
	< Get Response	
	< 00 C0 00 00 10h	
R || RND_T >		
	Authenticate (R, RND_T) >	
	80 82 00 00 10h > R > RND_T >	1. Compute Ks
		K_{sL} = 3DES (3DES (RND_C, K_C), K_T)
		2. R_1 = 3DES (RND_T, K_S)
		3. Get Response Block = R_1
		< 61 08h
	Get Response >	
	80 C0 00 00 08h >	
		< R_1

SAM	Terminal	ACOS
	< Verify ACOS Authentication < 80 7A 00 00 08h < R1	
Check if R_1 == DES (RND_T, K_S)		
If OK, 90 00h >	Proceed if both Card and Terminal are Authenticated	

Figure 17: Mutual Authentication with ACOS3 Cards

10.3. Submit PIN Enciphered

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

SAM	Terminal	ACOS3
	Get Card Serial Number >	
		< Serial Number
	< Use SAM Master Key$_I$ to compute Secret Code < Diversify (@K_{MI}, Serial Number) < 80 72 01h #K_{MI} 08h < Serial Number	
Compute SC using K_{MI} and Serial Number		
	< Encrypt command < 80 74 00 00 00 00h	
Compute R = ENC(SC, KS) 61 08h >		
	< Get Response < 00 C0 00 00 08h	
R >		
	Submit encrypted PIN > 80 20 06 00 08h > R >	
		< 90 00h

Figure 18: Submit Enciphered PIN for ACOS3

10.4. Change PIN Enciphered

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

SAM	Terminal	ACOS3
	Ask cardholder for Current PIN	
	Perform Submit Pin Enciphered of the previous section	
	Ask cardholder for new PIN	
Compute R = DEC (new PIN, K_S) Get Response Block = R 61 08h >	< Decrypt (@K_S, new PIN) < 80 76 00 01 08h < newPIN	
R >	< Get Response < 00 C0 00 00 08h	
	Change PIN (R) > 80 24 00 00 08h > R >	< 90 00h

Figure 19: Change Enciphered PIN for ACOS3

10.5. Secure Messaging Computation for ACOS3

Secure Messaging is supported by ACOS3 version 1.07 and later. 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 later
	Terminal wish to send to ACOS3: 80 INS P1 P2 P3 <Cmd Data>	
	Encrypt command data Set Seq# = 00 00 00 00 00 00 FF FFh AND RND_C + 1 Load Key Data to Load CBC Initial Vector. < 80 72 06 00 08 Seq#	Set Seq# = 00 00 00 00 00 00h FF FF AND RND_C + 1
Set Seq# as CBC initial vector 90 00h >	Encrypt command with CBC option: < 80 74 02 01 XXh <Cmd Data> Padding	
Compute: <Enc Data> = ENC_{CBC}(<Cmd Data> Padding, K_S, Seq#) 61 XXh >	Get Response < 00 C0 00 00 XXh	
<Enc Data> >		
	Compute MAC for secure messaging < Load Key Data to Load CBC Initial Vector. < 80 72 06 00 08h Seq#	
Set Seq# as CBC initial vector 90 00h >		
MAC_{Cmd} = $SIGN_{CBC}$(89 04 8Ch INS P1 P2 87 L_{87} Pi <Enc Data> Padding2, K_S, Seq#) 61 08h>	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 8Ch INS P1 P2 87 L_{87} Pi <Enc Data> Padding2	

SAM	Terminal	ACOS 3 v1.07 or later
	Get Response < 00 C0 00 00 08h	
MAC >		
	Send secure messaging command	
	8Ch INS P1 P2 P3* 87 L_{87} Pi <Enc Data> 8E 04h MAC_{Cmd}>	Verify MAC_{Cmd} Decrypt <Enc Data> Perform INS. Set Seq# ++ Compute MAC_{rsp} = $SIGN_{CBC}$(89 04h CLA* INS P1 P2 99 02h SW1 SW2 80 00 00 00 00 00h, K_S, Seq#)
	Get Response 80 C0 00 00 0Ah >	< 61 0Ah
		< 99 02h SW1 SW2 8E 04h MAC_{rsp}
	Verify MAC_{rsp} Load Key Data to Load CBC Initial Vector. < 80 72 06 00 08h ++Seq#	
Set Seq# as CBC initial vector 90 00h >		
Compute MAC_{rsp}* = $SIGN_{CBC}$(89 04h CLA* INS P1 P2 99 02h SW1 SW2 80 00 00 00 00 00h, K_S, Seq#) 61 08h>	Encrypt command with MAC option < 80 74 06 01 10 89 04h CLA* INS P1 P2 99 02h SW1 SW2 80 00 00 00 00 00h, Ks, Seq#) Get Response < 00 C0 00 00 08h	
MAC_{rsp}* >		
	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 20: Secure Messaging Computation for ACOS3 – Method 1

10.6. Secure Messaging Computation for ACOS3 – Method 2

In ACOS6-SAM revision 4.01 and higher, a more efficient method of computing ACOS3 secure messaging is available. This method lessens the workload of the terminal. As in the **Secure Messaging Computation for ACOS3**, 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 to ACOS3: 80 INS P1 P2 P3 <Cmd Data> The command is not be > 128 bytes	
	Encrypt command data < Encrypt, ACOS3 SM	
61 XXh >	< 80 74 08 01h Le [ACOS3 command]	
	< Get Response	
	< 00 C0 00 00 xxh	
[*SM ACOS3 Command*] >	SM ACOS3 Command] >	
		< 61 XXh
	Get Response >	
	00 C0 00 00 xxh >	< [*Encrypt Response and MAC*]
	< Decrypt, ACOS3 SM	
61 XXh >	< 80 76 08 01h Le [*Encrypt Response and MAC*]	
	< Get Response	
	< 00 C0 00 00 xxh	
[Response Data] 90 00h >		

Figure 21: Secure Messaging Computation for ACOS3 – Method 2

10.7. Inquire Account

SAM	Terminal	ACOS3
	Get Card Serial Number >	< Serial Number
Generate Certify Key Kacct $K_{ACCT}L$ = 3DES (SN, K_M) $K_{ACCT}R$ = 3DES (SN, K_M) 90 00h >	< Diversify (@K_M, Serial Number) < 80 72 02 #K_M 08h < Serial Number	
	Inquire Account (@K_{crt}) > 80 E4 02 00 04h > Ref >	< 61 19h
	Get Response > 80 C0 00 00 19h >	< MAC, TransType, Bal, ATREF, Max, $TTREF_C$, $TTREF_D$,
Compute MAC from K_{ACCT} and Datain, then compare with MAC in Datain R = 3MAC (Data, K_{ACCT}) If R == MAC 90 00h >	< Verify Inquire Account < 80 7C 00 00 1Dh < Ref, MAC, TransType, Balance, ATREF, Max, $TTREF_C$, $TTREF_D$ [4]	

Figure 22: Inquire Account of ACOS3

10.8. Debit

SAM	Terminal	ACOS3
	Get Card Serial Number >	
		< Serial Number
	< Diversify (@K_M, Serial Number)	
Generate Debit Key K_{ACCT} $K_{ACCT}L$ = 3DES (SN, K_M) $K_{ACCT}R$ = 3DES (SN, K_M) 90 00h >		
	Inquire Account (@K_D) >	
	80 E4 00 00 04h > Ref >	
		< 61 19h
	Get Response >	
	80 C0 00 00 19h >	
		< MAC, TransType, Bal, ATREF, Max, $TTREF_C$, $TTREF_D$,
	< Prepare ACOS Transaction	
	< 80 7E 00 E6 0x0Dh < Amount, $TTREF_D$, ATREF	
++ATREF Data = E6, Amount, $TTREF_D$, ATREF, 0, 0 R = 3MAC (K_{ACCT}, Data) R_1 = R || Amount || TTREFD Response Block = R_1 61 0Bh >		
	< Get Response	
	< 00 C0 00 00 0Bh	
R_1 >		
	Debit >	
	80 E6 01 00 0Bh > R_1 >	
		< Perform Debit and return Debit Certificate
		< 61 04h
	Get Response>	
	80 C0 00 00 04h>	
		< Debit Certificate
	< Verify Debit Certificate	
	< 00 70 00 00 04h < DC, AMT, New BAL, ATREF, $TTREF_D$	
Check if DC is correct 90 00h >		

Figure 23: Debit ACOS3 Purse

10.9. Credit

SAM	Terminal	ACOS3
	Get Card Serial Number >	
		< Serial Number
	< Diversify (@K_M, Serial Number)	
Generate Credit Key K_{ACCT} $K_{ACCT}L$ = 3DES (SN, K_M) $K_{ACCT}R$ = 3DES (SN, K_M,) 90 00h >		
	Inquire Account (@K_D) >	
	80 E4 01 00 04h > Ref >	
		< 61 19h
	Get Response >	
	80 C0 00 00 19h >	
		< MAC, TransType, Bal, ATREF, Max, $TTREF_C$, $TTREF_D$,
	< Prepare ACOS Transaction	
	< 00 7E 00 E2 0Dh < Amount, $TTREF_D$, ATREF	
++ATREF Data = E2, Amount, TTREFC, ATREF, 0, 0 R = MAC (Data, KACCT) R1 = R || Amount || $TTREF_C$ Get Response Block = R_1 61 0Bh >		
	< Get Response	
	< 00 C0 00 00 0Bh	
R_1 >		
	Credit >	
	80 E2 00 00 0Bh > R_1 >	
		< 90 00h
	Perform Verify Inquire Account with Credit Key and new amount	
	Inquire Account (@K_C) >	
	80 E4 01 00 04h > Ref >	
		< 61 19h
	Get Response >	
	80 C0 00 00 19h >	
		< MAC, TransType, Bal, ATREF, Max, $TTREF_C$, $TTREF_D$,

SAM	Terminal	ACOS3
	< Verify Inquire Account	
Compute MAC from K_{ACCT} and Datain, then compare with MAC in Datain	< 00 7C 00 00 1Dh < Ref, MAC, TransType, Balance, ATREF, Max, $TTREF_C$, $TTREF_D$ [4]	
Data = Ref, Trans Type, Bal, ATREF, 00, 00, $TTREF_C$, $TTREF_D$		
R = 3MAC (Data, K_{ACCT})		
If R == MAC		
90 00h >		

Figure 24: Credit ACOS3 Purse

10.10. 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 (see **Mutual Authentication with ACOS**) 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} 08h < Key version Number	
Generate Card Key		
90 00h >		
	< Generate Terminal Key using K_{MJ}	
	< 80 72 03 # K_{MJ} 08h < Key version Number	
Generate Terminal Key		
90 00h >		
	GET CHALLENGE >	
	00 84 00 00 08h >	< Generate and return RND_C
	< Pre ACOS Authenticate with RND_C	
Compute Session Key K_S	< 80 78 00 00 08h RND_C	
R = 3DES (RND_C, K_T) || RND_T		
Return R >		
	Mutual Authenticate >	
	00 82 K_C K_T 10h R	
		Compute Session Key K_S
		R_1 = 3DES (RND_T, K_S)
	< Verify ACOS Authentication	
	< 80 7A 00 00 08h R_1	< Return R_1
Check R_1		
Return OK >		
	GET CHALLENGE >	
	00 84 00 00 08h >	
	< Get Key from MSAM with Diversify	
	< 80 CA 01 K_I 10h RND_{SAM} || Key version Number	
Compute ENCkey		
Compute MAC		< Generate and return RND_{SAM}
Generate RND_{MSAM}		
RND_{MSAM} | ENCkey | MAC >		
	Set Key >	
	80 DA 00 K_J 1Ch RND_{MSAM} | ENCkey | MAC >	
		< 90 00h
	Repeat this procedure as necessary to inject other keys.	

Figure 25: Key Injection to ACOS6/ACOS6-SAM

11.0.SAM Scenarios for MIFARE Products

This section discusses the Transaction/Command flow for MIFARE based products, including MIFARE Ultralight C, MIFARE DESFire, MIFARE DESFire EV1/EV2/Light and MIFARE Plus.

11.1. MIFARE Ultralight C

11.1.1. 3 Pass Authentication with MIFARE 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
	< Generate bulk encryption Key using Master Key_J	
90 00h >	< 80 72 05 #K_{MJ} 00h	
	Start Authenticate >	
	1A 00h >	
		< ek(RndB)
	< Prepare Authentication	
	< 80 78 00 05 08h < ek(RndB)	
61 10h >		
	< Get Response	
	< 00 C0 00 00 10h	
ek(RndA + RndB') >		
	Authenticate (R, RNDt) >	
	AF ek(RndA + RndB') >	
		< 00 ek(RndA') 90 00h
	< Verify Authentication	
	< 80 7A 00 01 08h < ek(RndA')	
Check RndA		
If OK, 90 00h >		
If fail, 69 82h >		

Figure 26: Mutual Authentication with MIFARE Ultralight C with Undiversified Key

11.1.2. 3 Pass Authentication with MIFARE Ultralight C with Diversified Key

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

SAM	Terminal	MIFARE UL-C
	Retrieve UL-C's UID	
		<- UID
	Use as diversification data	
	< Generate Terminal Key using Master Key$_J$	
	< 80 72 03h #K_{MJ} 08h < 8 byte diversification data	
90 00h >		
	Start Authenticate >	
	1A 00h >	
		< ek(RndB)
	< Prepare Authentication	
	< 80 78 00 01 08h < ek(RndB)	
61 10h >		
	< Get Response	
	< 00 C0 00 00 10h	
ek(RndA + RndB')>		
	Authenticate (R, RNDt) >	
	AF ek(RndA + RndB')>	
		< 00 ek(RndA') (90 00h)
	< Verify Authentication	
	< 80 7A 00 01 08h < ek(RndA')	
Check RndA If OK, 90 00h > If fail, 69 82h >		

Figure 27: Mutual Authentication with MIFARE Ultralight C with Diversified Key

11.2. MIFARE DESFire EV1

11.2.1. Mutual Authentication with Static Key

ACOS6-SAM	Terminal	MIFARE DESFire
	< Generate bulk encryption Key using Master Key$_J$	
90 00h >	< 80 72 05h #K$_{MJ}$ 00h	
	Start Authenticate >	
	[Authenticate ISO] [Key No.] >	
		< [More]**[Challenge]**
	< Prepare ACOS Authentication	
	< 80 78 00 05 08/10h **[Challenge]**	
61 XXh >		
	< Get Response	
	< 00 C0 00 00 xxh	
[Authenticate Data] >		
	Authenticate >	
	[More] **[Authenticate Data]**>	
		< **[Success status][Verify Data]**
	< Verify Authentication	
	< 80 7A 00 01 08/10h **[Verify Data]**	
If OK, 90 00h >		
If fail, 69 82h >		

Figure 28: Mutual Authentication ISO with MIFARE DESFire EV1 Card (Key Not Diversified)

11.2.2. Mutual Authentication with Diversified Key

ACOS6-SAM	Terminal	MIFARE DESFire
	Get UID >	
		< Return UID
	< Generate Terminal Key using Master Key_J	
90 00h >	< 80 72 03h #K_{MJ} 08h < Pad UID to 8 byte	
	Start Authenticate >	
	[Authenticate ISO] [Key No.] >	
		< [More]**[Challenge]**
	< Prepare ACOS Authentication	
	< 80 78 00 05 08/10h **[Challenge]**	
61 xxh >		
	< Get Response	
	< 00 C0 00 00 xxh	
[Authenticate Data] >		
	Authenticate >	
	[More] **[Authenticate Data]**>	
		< **[Success status] [Verify Data]**
	< Verify Authentication	
	< 80 7A 00 01 08/10h **[Verify Data]**	
If OK, 90 00h >		
If fail, 69 82h >		

Figure 29: Mutual Authentication with MIFARE DESFire EV1 Card (Key Diversified)

11.2.3. Key Injection

After Mutual Authenticate, the Card can do the Key injection to MIFARE DESFire EV1 Card.

ACOS6-SAM	Terminal	MIFARE DESFire
	Mutual Authenticate (DES Session Key)	
	< Get Key < 80 CA 81h #K_M Lc [DATA] DATA include: - [8-byte SN], - 00, Original Key is DESFire Default Key - [1-byte DESFire Key No.], - 00h – Key Version	
61 XXh >		
	< Get Response < 00 C0 00 00 xxh	
[Encrypted Key Data] >		
	Change Key > [Change Key command] [Key No.] **[Encrypted Key Data]** >	< **[Success status] [CMAC]**
	< Encrypt – Check CMAC < 80 74 0F 01 01h **[Success status]**	
61 XXh >		
	< Get Response < 00 C0 00 00 xxh	
[CMAC] >		

Figure 30: 16-byte ISO Key Injection for MIFARE DESFire EV1 Card

11.2.4. Change Key

ACOS6-SAM	Terminal	MIFARE DESFire
	Mutual Authenticate (DES Session Key)	
	< Get Key < 80 CA 81h #K_M Lc [DATA] DATA include: - [8-byte SN], - #K_c, Original Key - [1-byte DESFire Key No.], - 00h – Key Version	
61 XXh >		
	< Get Response < 00 C0 00 00 xxh	
[Encrypted Key Data] >		
	Change Key > [Change Key command] [Key No.] **[Encrypted Key Data]** >	< **[Success status]** **[CMAC]**
	< Encrypt – Check CMAC < 80 74 0F 01 01h **[Success status]**	
61 XXh >		
	< Get Response < 00 C0 00 00 xxh	
[CMAC] >		

Figure 31: Change Key for MIFARE DESFire EV1 Card

11.2.5. Read Data

11.2.5.1. Full Encryption

ACOS6-SAM	Terminal	MIFARE DESFire
	Mutual Authenticate (DES Session Key)	
	Read Data >	
	[Read Data cmd and Para] >	
		< **[Success status]** **[Encrypted Data]**
	< Decrypt < 80 76 4D 01h Lc [DATA] DATA include: - Length of Data to Read - Length of Command and Parameter - **[Read Data cmd and Para]**, - **[Encrypted Data]**	
61 xxh >		
	< Get Response < 00 C0 00 00 xxh	
Plain Data >		

Figure 32: Read Full Encryption Data for MIFARE DESFire EV1 Card after Authenticate ISO

11.2.5.2. CMAC

ACOS6-SAM	Terminal	MIFARE DESFire
	Mutual Authenticate (DES Session Key)	
	< Encrypt – Cal CMAC < 80 74 0F 01h Lc **[Read Data cmd and Para]**	
61 xxh >		
	Read Data >	
	[Read Data cmd and Para] >	
		< [Status] **[Data]** **[CMAC]**
	< Decrypt < 80 74 0F 01h Lc **[Data]**	
61 xxh >		
	< Get Response < 00 C0 00 00 xxh	
[CMAC] >		

Figure 33: Read Data (CMAC) for MIFARE DESFire EV1 Card after Authenticate ISO

11.2.6. Write Data

11.2.6.1. Fully Encryption

ACOS6-SAM	Terminal	MIFARE DESFire
	Mutual Authenticate (DES Session Key)	
	< Encrypt 80 74 4D 01 Lc [DATA] - Length of Data to Write - Length of Command and Parameter - **[Write Data cmd and Para]**, - [Plain Data]	
61 xx >		
	< Get Response < 00 C0 00 00 xx	
[Write Data APDU] >		
	Write Data > **[Write Data APDU]** >	< **[Success status]** **[CMAC]**
	< Encrypt – Check CMAC < 80 74 0F 01 01 **[Success status]**	
61 XX >		
	< Get Response < 00 C0 00 00 xx	
[CMAC] >		

Figure 34: Write Data (Fully Encryption) for MIFARE DESFire EV1 Card

11.2.6.2. CMAC

ACOS6-SAM	Terminal	MIFARE DESFire
	Mutual Authenticate (DES Session Key)	
	< Encrypt – Cal CMAC < 80 74 0F 01h Lc **[Write Data cmd, Para & Data]**	
61 xxh >		
	< Get Response < 00 C0 00 00 xxh	
[CMAC1] >		
	Write Data > **[Write Data cmd, Para & Data][CMAC1]** >	
		< [Status] **[CMAC2]**
	< Encrypt – Cal CMAC < 80 74 0F 01h Le **[Status]**	
61 xxh >		

ACOS6-SAM	Terminal	MIFARE DESFire
	< Get Response < 00 C0 00 00 xxh	
[CMAC2] >		

Figure 35: Write Data (CMAC) for MIFARE DESFire EV1 Card after Authenticate ISO

11.3. MIFARE DESFire

11.3.1. Mutual Authentication with Static Key

ACOS6-SAM	Terminal	MIFARE DESFire
	< Generate bulk encryption Key using Master Key$_J$	
90 00h >	< 80 72 05h #K$_{MJ}$ 00h	
	Start Authenticate >	
	[Authenticate] [Key No.] >	
		< [More] **[Challenge]**
	< Prepare ACOS Authentication	
	< 80 78 81 05 08h **[Challenge]**	
61 xxh >		
	< Get Response	
	< 00 C0 00 00 xxh	
[Authenticate Data] >		
	Authenticate >	
	[More] **[Authenticate Data]**>	
		< **[Success status]** **[Verify Data]**
	< Verify Authentication	
	< 80 7A 00 01 08h **[Verify Data]**	
If OK, 90 00h >		
If fail, 69 82h >		

Figure 36: Mutual Authentication with MIFARE DESFire Card (Key Not Diversified)

11.3.2. Mutual Authentication with Diversified Key

ACOS6-SAM	Terminal	MIFARE DESFire
	Get UID >	
		< Return UID
	< Generate Terminal Key using Master Key$_J$	
90 00h >	< 80 72 03h #K$_{MJ}$ 08h < Pad UID to 8 byte	
	Start Authenticate >	
	[Authenticate] [Key No.] >	
		< [More]**[Challenge]**
	< Prepare ACOS Authentication	
	< 80 78 81 05 08h **[Challenge]**	
61 xxh >		
	< Get Response	
	< 00 C0 00 00 xxh	
[Authenticate Data] >		
	Authenticate >	
	[More] **[Authenticate Data]**>	
		< **[Success status][Verify Data]**
	< Verify Authentication	
	< 80 7A 00 01 08h **[Verify Data]**	
If OK, 90 00h >		
If fail, 69 82h >		

Figure 37: Mutual Authentication with MIFARE DESFire Card (Key Diversified)

11.3.3. Key Injection

After Mutual Authenticate, the Card can do the Key injection to MIFARE DESFire Card.

ACOS6-SAM	Terminal	MIFARE DESFire
	Mutual Authenticate (DES Session Key)	
	< Get Key < 80 CA 80h #K_M Lc [DATA] DATA include: - [8-byte SN], - 00h, Original Key is DESFire Default Key - [1-byte DESFire Key No.], - 00h – Key Version	
61 XXh >		
	< Get Response < 00 C0 00 00 xxh	
[Encrypted Key Data] >		
	Change Key >	
	[Change Key command] [Key No.] **[Encrypted Key Data]** >	
		< **[Success status]**

Figure 38: 16-byte ISO Key Injection for MIFARE DESFire Card

11.3.4. Change Key

ACOS6-SAM	Terminal	MIFARE DESFire
	Mutual Authenticate (DES Session Key)	
	< Get Key < 80 CA 80h #K_M Lc [DATA] DATA include: - [8-byte SN], - #K_C, Original Key - [1-byte DESFire Key No.], - 00h – Key Version	
61 XXh >		
	< Get Response < 00 C0 00 00 xxh	
[Encrypted Key Data] >		
	Change Key > [Change Key command] [Key No.] **[Encrypted Key Data]** >	
		< **[Success status]**

Figure 39: Change Key for MIFARE DESFire Card

11.3.5. Read Data

11.3.5.1. Full Encryption

ACOS6-SAM	Terminal	MIFARE DESFire
	Mutual Authenticate (DES Session Key)	
	Read Data >	
	[Read Data cmd and Para] >	
		< [Success status] **[Encrypted Data]**
	< Decrypt < 80 76 4B 01h Lc [DATA] DATA include: - Length of Data to Read - Length of Command and Parameter - **[Read Data cmd and Para]**, - **[Encrypted Data]**	
61 xxh >		
	< Get Response < 00 C0 00 00 xxh	
Plain Data >		

Figure 40: Read Full Encryption Data for MIFARE DESFire Card after 'Authenticate ISO'

11.3.5.2. MAC

ACOS6-SAM	Terminal	MIFARE DESFire
	Mutual Authenticate (DES Session Key)	
	<Initialize IV < 80 72 06 00 08 0000000000000000h	
90 00h >		
	Read Data >	
	[Read Data cmd and Para] >	
		< [Status] **[Data]** **[MAC_L]**
	< Decrypt < 80 74 06 01h Lc Padding (**[Data]**)	
61 xxh >		
	< Get Response < 00 C0 00 00 xxh	
[MAC_L] **[MAC_R]** >		

Figure 41: Read Data (CMAC) for MIFARE DESFire Card after 'Authenticate ISO'

11.3.6. Write Data

11.3.6.1. Full Encryption

ACOS6-SAM	Terminal	MIFARE DESFire
	Mutual Authenticate (DES Session Key)	
	<Initialize IV < 80 72 06 00 08 0000000000000000h	
90 00h >		
	< Encrypt 80 74 4B 01h Lc [DATA] - Length of Data to Write - Length of Command and Parameter - **[Write Data cmd and Para]**, - [Plain Data]	
61 xxh >		
	< Get Response < 00 C0 00 00 xxh	
[Write Data APDU] >		
	Write Data > **[Write Data APDU]** >	
		< [Success status]

Figure 42: Write Data (Full Encryption) for MIFARE DESFire Card

11.3.6.2. MAC

ACOS6-SAM	Terminal	MIFARE DESFire
	Mutual Authenticate (DES Session Key)	
	<Initialize IV < 80 72 06 00 08 0000000000000000h	
90 00h >		
	< Encrypt – Cal MAC < 80 74 06 01h Lc **[Write Data cmd, Para & Data]**	
61 xxh>		
	< Get Response < 00 C0 00 00 xxh	
[MAC_L] [MAC_R] >		
	Write Data > **[Write Data cmd, Para & Data]** **[MAC_L]** >	
		< [Status]

Figure 43: Write Data (MAC) for DESFire Card after 'Authenticate ISO'

11.4. MIFARE Plus

11.4.1. Mutual Authentication

11.4.1.1. Authenticate Switch Level key

To switch from SL1 to SL2 or SL3, This process can be completed in ACOS6-SAM with the relevant SL2 or SL3 switch key.

ACOS6-SAM	Terminal	MIFARE Plus
	Get SN >	
		< Return SN
	< Generate bulk encryption key using master key, or generate terminal key using master key.	
	< 80 72 05h #Kx 00h or	
	< 80 72 03h #Kx 08h < 8 bytes SN >	
90 00h >		
	Start Authenticate >	
	70 ||keyBNr ||lenCap || PCDcap2 >	
		< SC||E(RndB)
	< Prepare Authentication	
	< 80 78 03 02 10h < E(RndB) >	
61 20h >		
	< Get Response	
	< 00 C0 00 00 20h	
E(RndA + RndB') >		
	Authenticate >	
	72 || E(RndA + RndB'))>	
		< SC||verify data
	< Verify Authentication	
Check RndA	< 80 7A 00 02 20h < verify data >	
If OK, 90 00h >		
If fail, 69 82h >		

Figure 44: MIFARE Plus Authentication to switch from SL1 to SL2 or SL3

11.4.1.2. Authenticate in Security Level 2

ACOS6-SAM	Terminal	MIFARE Plus
	Get SN >	
		< Return SN
	< Generate bulk encryption key using master key, or generate terminal key using master key.	
90 00h >	< 80 72 05h #Kx 00h or < 80 72 03h #Kx 08h < 8 bytes SN >	
	Start Authenticate >	
	76 || keyBNr >	
		< SC|| E(RndB)
	< Prepare Authentication	
61 20h >	< 80 78 03 03 10h < E(RndB) >	
	< Get Response	
E(RndA + RndB') >	< 00 C0 00 00 20h	
	Authenticate >	
	72 || E(RndA + RndB')>	
		< SC|| E(RndA')
	< Verify Authentication	
61 06h >	< 80 7A 00 02 10h < E(RndA') >	
	< Get Response	
Crypto session key>	< 00 C0 00 00 06h	
	New crypto1 key = session key XOR MF crypto key	
	MF crypto1 key authenticate >	
		< SC

Figure 45: MIFARE Plus Authentication in Security Level 2

11.4.1.3. Authenticate in Security Level 3 (First Authentication)

ACOS6-SAM	Terminal	MIFARE Plus
	Get SN >	
		< Return SN
	< Generate bulk encryption key using master key, or generate terminal key using master key.	
90 00h >	< 80 72 05h #Kx 00h or < 80 72 0h3 #Kx 08h < 8 bytes SN >	
	Start Authenticate >	
	70 ||keyBNr || lenCap || PCDcap2 >	
		< SC || E(RndB)
	< Prepare Authentication	
61 20h >	< 80 78 03 02 10h < E(RndB) >	

ACOS6-SAM	Terminal	MIFARE Plus
	< Get response	
E(RndA + RndB') >	< 00 C0 00 00 20h	
	Authenticate >	
	72 || E(RndA + RndB')>	< SC || verify data
Check RndA	< Verify Authentication	
If OK, 90 00h >	< 80 7A 00 02 20h < Verify Data >	
If fail, 69 82h >		

Figure 46: MIFARE Plus First Authentication in Security Level 3

11.4.1.4. Authenticate in Security Level 3 (Following Authentication)

ACOS6-SAM	Terminal	MIFARE Plus
	First Authentication in secure level3	
	Start Authenticate >	
	76 || keyBNr >	< SC|| E(RndB)
	< Prepare Authentication	
61 20h >	< 80 78 03 04 10h < E(RndB) >	
	< Get Response	
E(RndA + RndB') >	< 00 C0 00 00 20h	
	Authenticate >	
	72 || (E(RndA + RndB')) >	< SC || E(RndA')
Check RndA	< Verify Authentication	
If OK, 90 00h >	< 80 7A 00 02 10h < E(RndA') >	
If fail, 69 82h >		

Figure 47: MIFARE Plus Following Authentication in Security Level 3

11.4.2. MIFARE Plus Command Encryption/MAC

11.4.2.1. Read Command

ACOS6-SAM	Terminal	Mifare Plus
	Mutual Authentication: Generate Session Key K_{ENC} and K_{MAC}	
	<Encrypt command, Process value command < 80 74 A1 01 04h Code ||BNr||Ext	
61 xxh >		
	< Get Response < 00 C0 00 00 xxh	
[Code ||BNr||Ext|| MAC(optional)]>		
	Read command >	
		<SC|| Block Data||RMAC(optional)
	<Encrypt command, Process the response <80 74 A3 01h Lc Block Data	
RMAC exist, 61 08h, else, 90 00h >		
	If RMAC exist, please send command as follows <Get Response <00 C0 00 00 08h	
RMAC >	Compare with RMAC in MFP response	

Figure 48: MIFARE Plus Read Command Encryption/MAC

11.4.2.2. Write Command

ACOS6-SAM	Terminal	MIFARE Plus
	Mutual authentication: Generate session key K_{ENC} and K_{MAC}	
	<Encrypt command, Process value command < 80 74 A1 01 04h Code||BNr||Write Data	
61 xxh>		
	< Get Response <00 C0 00 00 xxh	
[Code||BNr||Write Data (Encrypt Optional)|| MAC(optional)]>	Write command	
	Write command >	
		<SC|| RMAC(optional)

ACOS6-SAM	Terminal	MIFARE Plus
	<Encrypt command, Process the response	
RMAC exist, 61 08h, else, 90 00h>	<80 74 A3 01 00h	
	If RMAC exist, please send command as follows	
	<Get Response	
	<00 C0 00 00 08h	
RMAC>	Compare with RMAC in MFP response	

Figure 49: MIFARE Plus Write Command Encryption/MAC

11.4.2.3. Value Command

ACOS6-SAM	Terminal	MIFARE Plus
	Mutual authentication: Generate session key K_{ENC} and K_{MAC}	
	<Encrypt command, Process value command	
61 xxh>	< 80 74 A1 01 04h Code||BNrs||Value	
	< Get Response	
[Code||BNrs|| Encrypt Value || MAC(optional)]>	<00 C0 00 00 xxh Value command	
	Value command >	
		<SC|| RMAC(optional)
	<Encrypt command, Process the response	
RMAC exist, 61 08h, else, 90 00h>	<80 74 A3 01 00h	
	If RMAC exist, please send command as follows	
	<Get Response	
	<00 C0 00 00 08h	
RMAC>	Compare with RMAC in MFP response	

Figure 50: MIFARE Plus Value Command Encryption/MAC

11.4.3. MIFARE Plus Response Data Decryption

11.4.3.1. Read Command with Encrypted Response

ACOS6-SAM	Terminal	MIFARE Plus
	Mutual authentication: Generate session key K_{ENC} and K_{MAC}	
	<Encrypt command, Process value command < 80 74 A1 01 04h Code ||BNr||Ext	
61 xxh >		
	< Get Response <00 C0 00 00 xxh	
[Code ||BNr||Ext|| MAC(optional)]>	Read command	
	Read command >	<SC|| **Encrypted Block Data** ||RMAC(optional)
	< Decrypt command, Process the response <80 76 A5 01h Lc **[Encrypted Block Data]**	
61 xxh >		
	<Get Response <00 C0 00 00 xxh	
Plain Data >		

Figure 51: MIFARE Plus Read Command Decrypting Encrypted Response Data

11.4.4. Change Key

ACOS6-SAM	Terminal	MIFARE Plus
	Mutual authentication: Generate session key K_{ENC} and K_{MAC}	
	< Get Key < 80 CA 83h KeyID 0Bh [DATA] DATA include: - Serial Number for target card (8 Byte) - Write command A1 (A0 or A1) - BNr (2 Byte)	
61 xxh >		
	< Get Response < 00 C0 00 00 xxh	
[Write**Command] [BNr] [Encrypted][MAC** (optional)**] >**		
	Write command	
	Write command >	
		<[Success status][RMAC(optional**)]**
	< Verify RMAC **(**optional**)**	
RMAC exist, 61 08h, else, 90 00h>	< 80 74 A3 01 00h If RMAC exist, please send command as follows <Get Response <00 C0 00 00 08h	
RMAC >	Compare with RMAC in MFP response	

Figure 52: MIFARE Plus Change Key Command Flow

11.5. MIFARE Desfire EV2

11.5.1. Mutual Authentication

11.5.1.1. Authenticate EV2 First

ACOS6-SAM	Terminal	MIFARE Plus
	Get SN >	
		< Return SN
	< Generate bulk encryption key using master key, or generate terminal key using master key.	
90 00h >	< 80 72 05h #Kx 00h or < 80 72 03h #Kx 08h < 8 bytes SN >	
	Start Authenticate EV2 First >	
	71 ||keyNo || lenCap || PCDcap2 >	< AFh || E(RndB)
	< Prepare Authentication	
61 20h >	< 80 78 03 06 10h < E(RndB) >	
	< Get response	
E(RndA + RndB') >	< 00 C0 00 00 20h	
	Authenticate >	
	AFh || E(RndA + RndB')>	< 00h || verify data
Check RndA If OK, 90 00h > If fail, 69 82h >	< Verify Authentication	
	< 80 7A 00 02 20h < Verify Data >	

Figure 53: MIFARE Desfire EV2 First

11.5.1.2. Authenticate EV2NonFirst

ACOS6-SAM	Terminal	MIFARE Plus
	Start Authenticate EV2NonFirst> 77|| keyNo >	< AFh || E(RndB)
61 20h >	< Prepare Authentication < 80 78 03 06 10h < E(RndB) >	
E(RndA + RndB') >	< Get Response < 00 C0 00 00 20h	
	Authenticate > AFh || (E(RndA + RndB')) >	< 00h || E(RndA')
Check RndA If OK, 90 00h > If fail, 69 82h >	< Verify Authentication < 80 7A 00 02 10h < E(RndA') >	

Figure 54: MIFARE Desfire EV2NonFirst

11.5.2. Key Injection

After Mutual Authenticate, the Card can do the Key injection to MIFARE DESFire EV2 Card.

ACOS6-SAM	Terminal	MIFARE DESFire
	AuthenticateEV2First Or AuthenticateEV2NonFirst	
61 XXh >	< Get Key < 80 CA 81h #K_M Lc [DATA] DATA include: - [8-byte SN], - 00, Original Key is DESFire Default Key - [1-byte DESFire Key No.], - 00h – Key Version - C6h (Optional) - 00h KeySetNo (Optional)	
[ChangeKeyEV2 command And Para] **[Encrypted Key Data]**>	< Get Response < 00 C0 00 00 xxh	
	ChangeKeyEV2 > [ChangeKeyEV2 command And Para] **[Encrypted Key Data]**>	< **[Success status] [CMAC]**
90 00h >	< Encrypt – Check CMAC < 80 74 8F 01 09h **[Success status] [CMAC]**	

Figure 55: 16-byte AES Key Injection for MIFARE DESFire EV2 Card

11.5.3. Change Key

ACOS6-SAM	Terminal	MIFARE DESFire
	AuthenticateEV2First Or AuthenticateEV2NonFirst	
61 XXh >	< Get Key < 80 CA 81h #K_M Lc [DATA] DATA include: - [8-byte SN], - #K_c, Original Key - [1-byte DESFire Key No.], - 00h – Key Version - C6h (Optional) - 00h KeySetNo (Optional)	
[ChangeKeyEV2 command And Para] **[Encrypted Key Data]** >	< Get Response < 00 C0 00 00 xxh	
	ChangeKeyEV2 > [ChangeKeyEV2 command And Para] **[Encrypted Key Data]** >	< **[Success status] [CMAC]**
90 00h >	< Encrypt – Check CMAC < 80 74 8F 01 09h **[Success status] [CMAC]**	

Figure 56: Change Key for MIFARE DESFire EV2 Card

11.5.4. Read Data

11.5.4.1. Full Encryption

ACOS6-SAM	Terminal	MIFARE DESFire
	AuthenticateEV2First Or AuthenticateEV2NonFirst	
	< Encrypt – Cal CMAC < 80 74 8F 01h Lc **[Read Data cmd and Para]**	
61 xxh		
	< Get Response < 00 C0 00 00 xxh	
[Read Data cmd and Para] [CMAC]>		
	Read Data > **[Read Data cmd and Para] [CMAC]** >	
		< **[Success status] [Encrypted Data] [CMAC]**
	< Decrypt and Check the CMAC < 80 76 CD 01h Lc [DATA] DATA include: **[Success status]** **[Encrypted Data]** **[CMAC]**	
61 xxh >		
	< Get Response < 00 C0 00 00 xxh	
Plain Data >		

Figure 57: Read Full Encryption Data for MIFARE DESFire EV2 Card after Authenticate EV2

11.5.4.2. CMAC

ACOS6-SAM	Terminal	MIFARE DESFire
	AuthenticateEV2First Or AuthenticateEV2NonFirst	
	< Encrypt – Cal CMAC < 80 74 8F 01h Lc **[Read Data cmd and Para]**	
61 xxh >		
	< Get Response < 00 C0 00 00 xxh	
[Read Data cmd and Para] [CMAC]>		

ACOS6-SAM	Terminal	MIFARE DESFire
	Read Data >	
	[Read Data cmd and Para] [CMAC]>	
		< **[Success status]** **[Data] [CMAC]**
	< Check the CMAC	
	< 80 74 8F 01h Lc **[Data]**	
	DATA include:	
	[Success status]	
	[Data]	
	[CMAC]	
90 00h >		

Figure 58: Read Data (CMAC) for MIFARE DESFire EV2 Card after Authenticate EV2

11.5.5. Write Data

11.5.5.1. Fully Encryption

ACOS6-SAM	Terminal	MIFARE DESFire
	AuthenticateEV2First Or AuthenticateEV2NonFirst	
	< Encrypt 80 74 CD 01 Lc [Plain Data]	
61 xx >		
	< Get Response < 00 C0 00 00 xx	
[Encrypted Data] >		
	< Encrypt – Cal CMAC < 80 74 8F 01h Lc **[Write Data cmd and Para]** **[Encrypted Data]**	
61 xxh >		
	< Get Response < 00 C0 00 00 xx	
[Write Data cmd and Para] **[Encrypted Data][CMAC]**>		
	Write Data > **[Write Data cmd and Para]** **[Encrypted Data][CMAC]** >	< **[Success status]** **[CMAC]**
	< Encrypt – Check CMAC < 80 74 8F 01 09 **[Success status]** **[CMAC]**	
90 00h >		

Figure 59: Write Data (Fully Encryption) for MIFARE DESFire EV2 Card after Authenticate EV2

11.5.5.2. CMAC

ACOS6-SAM	Terminal	MIFARE DESFire
	AuthenticateEV2First Or AuthenticateEV2NonFirst	
	< Encrypt – Cal CMAC < 80 74 8F 01h Lc **[Write Data cmd, Para & Data]**	
61 xxh >		
	< Get Response < 00 C0 00 00 xxh	
[Write Data cmd, Para & Data] [CMAC] >		

ACOS6-SAM	Terminal	MIFARE DESFire
	Write Data >	
	[Write Data cmd, Para & Data][CMAC] >	
		< [Status] **[CMAC]**
	< Encrypt – Check the CMAC	
	< 80 74 8F 01 09 [Status] **[CMAC]**	
90 00h>		

Figure 60: Write Data (CMAC) for MIFARE DESFire EV2 Card after Authenticate EV2

11.5.6. Credit

11.5.6.1. Fully Encryption

ACOS6-SAM	Terminal	MIFARE DESFire
	AuthenticateEV2First Or AuthenticateEV2NonFirst	
	< Encrypt 80 74 CD 01 Lc [Plain Data]	
61 xx >		
	< Get Response < 00 C0 00 00 xx	
[Encrypted Data] >		
	< Encrypt – Cal CMAC < 80 74 8F 01h Lc **[Credit cmd and Para] [Encrypted Data]**	
61 xxh >		
	< Get Response < 00 C0 00 00 xx	
[Credit cmd and Para] [Encrypted Data][CMAC]>		
	Credit > **[Credit cmd and Para] [Encrypted Data][CMAC]** >	< **[Success status] [CMAC]**
	< Encrypt – Check CMAC < 80 74 8F 01 09 **[Success status] [CMAC]**	
90 00h >		

Figure 61: Credit (Fully Encryption) for MIFARE DESFire EV2 Card after Authenticate EV2

11.5.7. Debit

11.5.7.1. Fully Encryption

ACOS6-SAM	Terminal	MIFARE DESFire
	AuthenticateEV2First Or AuthenticateEV2NonFirst	
	< Encrypt 80 74 CD 01 Lc [Plain Data]	
61 xx >		
	< Get Response < 00 C0 00 00 xx	
[Encrypted Data] >		
	< Encrypt – Cal CMAC < 80 74 8F 01h Lc **[Debit cmd and Para] [Encrypted Data]**	
61 xxh >		
	< Get Response < 00 C0 00 00 xx	
[Debit cmd and Para] [Encrypted Data][CMAC]>		
	Debit > **[Debit cmd and Para] [Encrypted Data][CMAC]** >	< **[Success status] [CMAC]**
	< Encrypt – Check CMAC < 80 74 8F 01 09 **[Success status] [CMAC]**	
90 00h >		

Figure 62: Debit (Fully Encryption) for MIFARE DESFire EV2 Card after Authenticate EV2

11.6. MIFARE Desfire Light

11.6.1. Mutual Authentication

11.6.1.1. Authenticate EV2 First

ACOS6-SAM	Terminal	MIFARE Plus
	Get SN >	
		< Return SN
	< Generate bulk encryption key using master key, or generate terminal key using master key.	
90 00h >	< 80 72 05h #Kx 00h or < 80 72 03h #Kx 08h < 8 bytes SN >	
	Start Authenticate EV2 First >	
	90h||71h ||00h||00h||xx||keyNo || lenCap || PCDcap2||00h >	< E(RndB)||91h||AFh
61 20h >	< Prepare Authentication < 80 78 03 06 10h < E(RndB) >	
E(RndA + RndB') >	< Get response < 00 C0 00 00 20h	
	Authenticate >	
	90h||AFh ||00h||00h||20h|| E(RndA + RndB') ||00h >	< verify data||91h||00h
Check RndA If OK, 90 00h > If fail, 69 82h >	< Verify Authentication < 80 7A 00 02 20h < Verify Data >	

Figure 63: MIFARE Desfire Light AuthenticateEV2First

11.6.1.2. Authenticate EV2NonFirst

ACOS6-SAM	Terminal	MIFARE Plus
	Start Authenticate EV2NonFirst> 90h||77h ||00h||00h||01h || keyNo||00h >	< E(RndB) ||91h||AFh
61 20h >	< Prepare Authentication < 80 78 03 06 10h < E(RndB) >	
E(RndA + RndB') >	< Get Response < 00 C0 00 00 20h	
	Authenticate > 90h||AFh ||00h||00h||20h|| (E(RndA + RndB')) ||00h >	< E(RndA') ||91h||00h
Check RndA If OK, 90 00h > If fail, 69 82h >	< Verify Authentication < 80 7A 00 02 10h < E(RndA') >	

Figure 64: MIFARE Desfire Light AuthenticateEV2NonFirst

11.6.2. Key Injection

After Mutual Authenticate, the Card can do the Key injection to MIFARE DESFire Light Card.

ACOS6-SAM	Terminal	MIFARE DESFire
	AuthenticateEV2First Or AuthenticateEV2NonFirst	
	< Get Key < 80 CA 81h #K_M Lc [DATA] DATA include: - [8-byte SN], - 00, Original Key is DESFire Default Key - [1-byte DESFire Key No.], - 00h – Key Version	
61 XXh >		
	< Get Response < 00 C0 00 00 xxh	
C4h || [KeyNo] [Encrypted Key Data]>		
	ChangeKey > 90h||C4h ||00h||00h||29h|| [KeyNo] [Encrypted Key Data] ||00h >	< **[CMAC] ||91h||00h**
	< Encrypt – Check CMAC < 80 74 8F 01 09h **00h | | [CMAC]**	
90 00h >		

Figure 65: 16-byte AES Key Injection for MIFARE DESFire Light Card

11.6.3. Change Key

ACOS6-SAM	Terminal	MIFARE DESFire
	AuthenticateEV2First Or AuthenticateEV2NonFirst	
61 XXh >	< Get Key < 80 CA 81h #K_M Lc [DATA] DATA include: - [8-byte SN], - #K_c, Original Key - [1-byte DESFire Key No.], - 00h – Key Version	
C4h || [KeyNo] [Encrypted Key Data] >	< Get Response < 00 C0 00 00 xxh	
	ChangeKey > 90h||C4h ||00h||00h||29h|| [KeyNo] [Encrypted Key Data] ||00h >	< **[CMAC] ||91h||00h**
90 00h >	< Encrypt – Check CMAC < 80 74 8F 01 09h **00h || [CMAC]**	

Figure 66: Change Key for MIFARE DESFire Light Card

11.6.4. Read Data

11.6.4.1. Full Encryption

ACOS6-SAM	Terminal	MIFARE DESFire
	AuthenticateEV2First Or AuthenticateEV2NonFirst	
	< Encrypt – Cal CMAC < 80 74 8F 01h Lc || ADh || [FileNo] [Offset] [Length]	
61 xxh		
	< Get Response < 00 C0 00 00 xxh	
ADh || [FileNo] [Offset] [Length] [CMAC]>		
	Read Data > 90h||ADh ||00h||00h||0Fh||[FileNo] [Offset] [Length] [CMAC]||00h >	< **[Encrypted Data] [CMAC] ||91h||00h**
	< Decrypt and Check the CMAC < 80 76 CD 01h Lc [DATA] DATA include: **00h** **[Encrypted Data]** **[CMAC]**	
61 xxh >		
	< Get Response < 00 C0 00 00 xxh	
Plain Data >		

Figure 67: Read Full Encryption Data for MIFARE DESFire Light Card after Authenticate EV2

11.6.5. Write Data

11.6.5.1. Fully Encryption

ACOS6-SAM	Terminal	MIFARE DESFire
	AuthenticateEV2First Or AuthenticateEV2NonFirst	
	< Encrypt 80 74 CD 01 Lc [Plain Data]	
61 xx >		
	< Get Response < 00 C0 00 00 xx	
[Encrypted Data] >		
	< Encrypt – Cal CMAC < 80 74 8F 01h Lc || 8Dh || [FileNo] [Offset] [Length] **[Encrypted Data]**	
61 xxh >		
	< Get Response < 00 C0 00 00 xx	
8Dh || [FileNo] [Offset] [Length] **[Encrypted Data][CMAC]**>		
	Write Data > 90h||8Dh ||00h||00h||xxh||[FileNo] [Offset] [Length] **[Encrypted Data][CMAC]** ||00h >	**< [CMAC] ||91h||00h**
	< Encrypt – Check CMAC < 80 74 8F 01 09 **00h || [CMAC]**	
90 00h >		

Figure 68: Write Data (Fully Encryption) for MIFARE DESFire Light Card after Authenticate EV2

11.6.6. Credit

11.6.6.1. Fully Encryption

ACOS6-SAM	Terminal	MIFARE DESFire
	AuthenticateEV2First Or AuthenticateEV2NonFirst	
	< Encrypt 80 74 CD 01 04h [Plain Data]	
61 xx >		
	< Get Response < 00 C0 00 00 xx	
[Encrypted Data] >		
	< Encrypt – Cal CMAC < 80 74 8F 01h Lc || 0Ch || [FileNr] **[Encrypted Data]**	
61 xxh >		
	< Get Response < 00 C0 00 00 xx	
0Ch || [FileNr] **[Encrypted Data][CMAC]**>		
	Credit> 90h||0Ch ||00h||00h||xxh|| [FileNr] **[Encrypted Data][CMAC]** ||00h >	< **[CMAC] ||91h||00h**
	< Encrypt – Check CMAC < 80 74 8F 01 09 **00h || [CMAC]**	
90 00h >		

Figure 69: Credit (Fully Encryption) for MIFARE DESFire Light Card after Authenticate EV2

11.6.7. Debit

11.6.7.1. Fully Encryption

ACOS6-SAM	Terminal	MIFARE DESFire
	AuthenticateEV2First Or AuthenticateEV2NonFirst	
	< Encrypt 80 74 CD 01 Lc [Plain Data]	
61 xx >		
	< Get Response < 00 C0 00 00 xx	
[Encrypted Data] >		
	< Encrypt – Cal CMAC < 80 74 8F 01h Lc || DCh || [FileNr] **[Encrypted Data]**	
61 xxh >		
	< Get Response < 00 C0 00 00 xx	
DCh || [FileNr] **[Encrypted Data][CMAC]**>		
	Debit > 90h||DCh ||00h||00h||xxh|| [FileNr] **[Encrypted Data][CMAC]** ||00h >	< **[CMAC] ||91h||00h**
	< Encrypt – Check CMAC < 80 74 8F 01 09 **00h || [CMAC]**	
90 00h >		

Figure 70: Debit (Fully Encryption) for MIFARE DESFire Light Card after Authenticate EV2

12.0. Quick Start Guide

This Quick Start Guide was created to help application developers be familiarized with ACOS6-SAM. In order to use the SAM card, it has to be initialized first. The initialization involves creating a file system on the card and loading a Master Key set. Application developers will first have to be familiar with the ACOS6-SAM commands and file system to be able to design the file system and key set based on the application needs. Likewise, to help application developers shorten this process, this document aims to guide the user through the initialization and go through some of the authentication procedures.

12.1. ACOS3 Client Card Sample

The included script works with ACS Script Tools. 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.

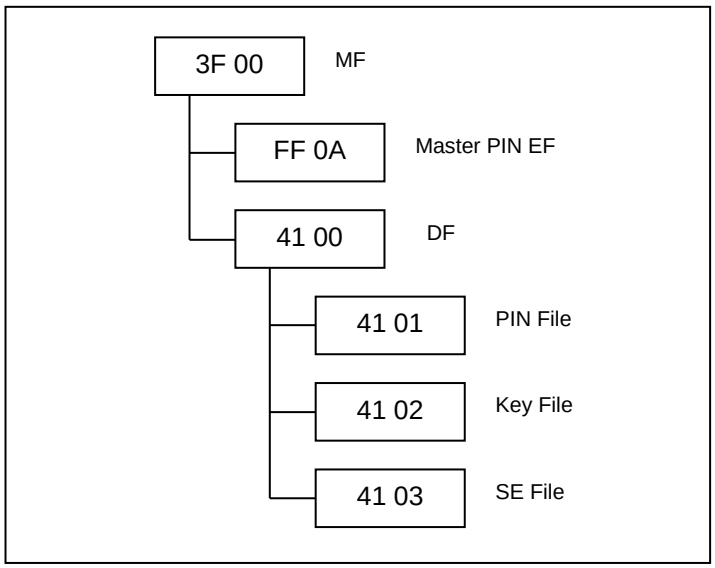


Figure 71: Sample Script File System of SAM

The script listed below will generate the file system on a blank card. It is important to note that this script can be run only once. Once the MF is created, it cannot be removed unless the CLEAR CARD command is called and the card life cycle fuse (in **Card Life Cycle States**) 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 three 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)
```

12.1.1. 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 a 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
```

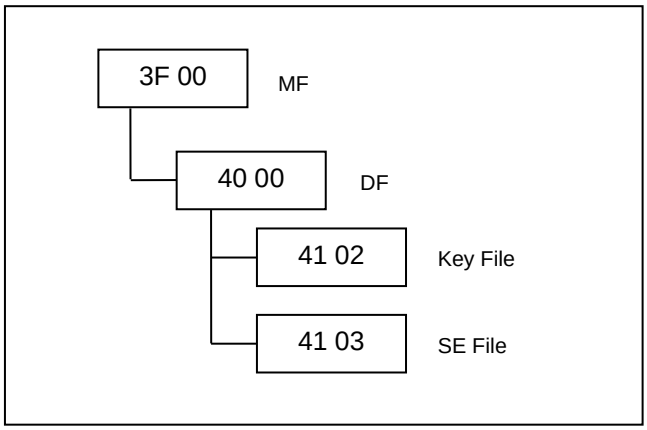
12.1.2. Example of Mutual Authentication with SAM

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 02h 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
```

12.2. 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)
```

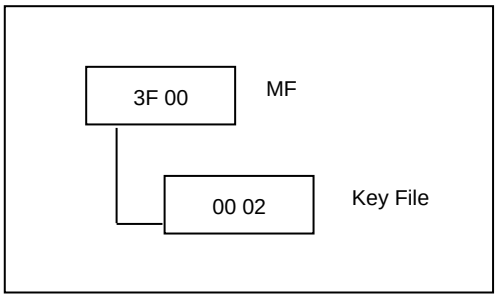
12.2.1. Example of Short Key External Authenticate

This demonstrates 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
```

12.3. 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 Ultralight 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 (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 Ultralight 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)
```

12.3.1. Example of Getting the Diversify Key

The following two commands get 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
```

12.3.2. Example of APDU flow of MIFARE Ultralight C Authenticate with static key

The following authentication step follows the scenario based on **3 Pass Authentication with MIFARE Ultralight C with Static Key**. This procedure is useful in initializing MIFARE Ultralight 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
```

12.3.3. Example of APDU Flow of MIFARE Ultralight C with Diversified Key

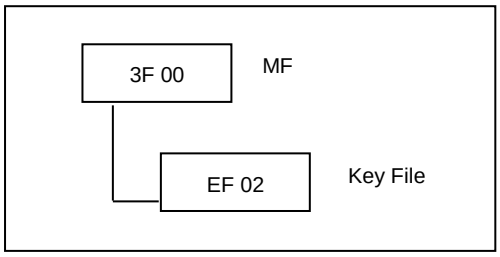
The following authentication step follows the scenario based on **3 Pass Authentication with MIFARE Ultralight C with Diversified Key**.

Note: *It is assumed that the key inside the MIFARE Ultralight C card is already diversified.*

```
; ACOS6-SAM Card - diversified terminal key local key 2 to RAM with a
diversification data (i.e. 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
```

12.4. MIFARE Plus Sample

The following initialization just creates one key file to demonstrate the MIFARE Plus:




```
;create MF
00 E0 00 00 12 62 10 82 02 3F FF 83 02 3F 00 84 06 A1 A2 A3 A4 A5 A6 (9000)

;create key file,EF2
00 E0 00 00 0D 62 0B 82 05 0C 00 00 16 03 83 02 EF 02 (9000)

;Append key
;ID:81, type:02,INTAUTH, info:FFFF, algo:03 AES, value: 16bytes FF
00 DC 01 04 15 81 02 FF FF 03 FF FF FF FF FF FF FF FF FF FF FF FF FF FF
FF (9000)

;Append key
;ID:82, type:04, Bulk key, info:non, algo:03 AES, value: 16bytes FF
00 DC 02 04 13 82 04 03 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
(9000)

;active file
00 44 00 00 02 3F 00 (9000)
```

12.4.1. Example of Mutual Authentication

The following example is for a MIFARE Plus Card that is in Security level 3, showing how the ACOS6-SAM interacts with a MIFARE Plus card to execute the first authentication and the following authentication. This procedure is useful in initializing MIFARE Plus cards with a default key.

```
;select MF
<SAM 00 A4 00 00 02 3F 00
>SAM 6120

<SAM 00 C0 00 00 20
>SAM 6F 1E 83 02 3F 00 88 01 00 8A 01 05 82 02 3F FF 8D 02 FF FF 84 06 A1
A2 A3 A4 A5 A6 8C 00 AB 00 9000

; ACOS6-SAM Card - load the bulk key to RAM
;diversify key ;no diversify
<SAM 80 72 05 02 00
>SAM 9000

;first authentication step 1
<Card 70 25 40 00
>Card 90 <ek(RndB)>

<SAM 80 78 03 02 10 <ek(RndB)>
>SAM 6120
<SAM 00 C0 00 00 20
>SAM <ek(RndA + RndB')> 9000
```

```
;first authentication step 2
<Card 72 <ek(RndA + RndB')>
>Card 90 <TI><ek(RndA')><PICCcap2><PCDcap2>

;Verify the card return value
<SAM 80 7A 00 02 20 <TI><ek(RndA')><PICCcap2><PCDcap2>
>SAM 9000

;following authentication step 1
<Card 76 25 40
>Card 90 <ek(RndB)>

<SAM 80 78 03 04 10 <ek(RndB)>
>SAM 6120
<SAM 00 C0 00 00 20
>SAM <ek(RndA + RndB')> 9000

;following authentication step 2
<Card 72 <ek(RndA + RndB')>
>Card 90 <ek(RndA')>

;Verify the card return value
<SAM 80 7A 00 02 10 <ek(RndA')>
>SAM 9000
```

12.4.2. Example of Change Key (or MIFARE Plus initialization)

The following example is for a MIFARE Plus Card that is in Security level 3, showing how the ACOS6-SAM interacts with a MIFARE Plus card to change the AES Key of MIFARE Plus card .This procedure is useful in initializing MIFARE Plus cards with a default key.

```
;select MF
<SAM 00 A4 00 00 02 3F 00
>SAM 6120

<SAM 00 C0 00 00 20
>SAM 6F 1E 83 02 3F 00 88 01 00 8A 01 05 82 02 3F FF 8D 02 FF FF 84 06 A1
A2 A3 A4 A5 A6 8C 00 AB 00 9000

; ACOS6-SAM Card - load the bulk key(ID:82) to RAM
;diversify key ;no diversify
<SAM 80 72 05 02 00
>SAM 9000

;first authentication step 1 authentication keyA
<Card 70 0A 40 00
>Card 90 <ek(RndB)>

<SAM 80 78 03 02 10 <ek(RndB)>
>SAM 6120
<SAM 00 C0 00 00 20
>SAM <ek(RndA + RndB')> 9000

;first authentication step 2
<Card 72 <ek(RndA + RndB')>
>Card 90 <TI><ek(RndA')><PICCcap2><PCDcap2>

;Verify the card return value
<SAM 80 7A 00 02 20 <TI><ek(RndA')><PICCcap2><PCDcap2>
>SAM 9000
```

```
;GET key
<SAM 80 CA 03 81 0B <8-byte SN> A0 0B 40
>SAM 611B
<SAM 00 C0 00 00 1B
>SAM A0 0B 40 <16-byte Diversified Key><8-byte MAC> 9000

;Change KeyB load the diversify key to Mifare Plus card
<Card A0 0B 40 <16-byte Diversified Key><8-byte MAC>
>Card 90

;Process the response
<SAM 80 74 A3 01 00
>SAM 9000
```

12.4.3. Example of Encrypted Writing

The following example is for a MIFARE Plus Card that is in Security level 3, showing how the ACOS6-SAM interacts with a MIFARE Plus card to Encrypted Writing data into MIFARE Plus card. This procedure is useful in initializing MIFARE Plus cards with a default key.

```
;select MF
<SAM 00 A4 00 00 02 3F 00
>SAM 6120

<SAM 00 C0 00 00 20
>SAM 6F 1E 83 02 3F 00 88 01 00 8A 01 05 82 02 3F FF 8D 02 FF FF 84 06 A1
A2 A3 A4 A5 A6 8C 00 AB 00 9000

; ACOS6-SAM Card - load the bulk key(ID:82) to RAM
;diversify key ;no diversify
<SAM 80 72 05 02 00
>SAM 9000

;first authentication step 1
<Card 70 0B 40 00
>Card 90 <ek(RndB)>

<SAM 80 78 03 02 10 <ek(RndB)>
>SAM 6120
<SAM 00 C0 00 00 20
>SAM <ek(RndA + RndB')> 9000

;first authentication step 2
<Card 72 <ek(RndA + RndB')>
>Card 90 <TI><ek(RndA')><PICCcap2><PCDcap2>

;Verify the card return value
<SAM 80 7A 00 02 20 <TI><ek(RndA')><PICCcap2><PCDcap2>
>SAM 9000

;write: command with enc and mac, response with none
<SAM 80 74 A1 01 13 A0 14 00 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
FF
>SAM 611B
<SAM 00 C0 00 00 1B
>SAM A0 14 00 <ek(16-byte data)><8-byte MAC> 9000

<Card A0 14 00 <ek(16-byte data)><8-byte MAC>
<Card 90

;Process the response
```

```
<SAM 80 74 A3 01 00
>SAM 9000

;write: command with enc and mac, response with mac
<SAM 80 74 A1 01 13 A1 14 00 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
FF
>SAM 611B
<SAM 00 C0 00 00 1B
>SAM A1 14 00 <ek(16-byte data)><8-byte MAC> 9000

<Card A1 14 00 <ek(16-byte data)><8-byte MAC>
<Card 90 <8-byte MAC>

;Process the response
<SAM 80 74 A3 01 00
>SAM 6108
<SAM C0 00 00 08
>SAM <8-byte MAC> 9000
```

12.4.4. Example of Encrypted Reading

The following example is for a MIFARE Plus Card that is in Security level 3, showing how the ACOS6-SAM interacts with a MIFARE Plus card to Read Encrypted data from MIFARE Plus card. This procedure is useful in initializing MIFARE Plus cards with a default key.

```
;select MF
<SAM 00 A4 00 00 02 3F 00
>SAM 6120

<SAM 00 C0 00 00 20
>SAM 6F 1E 83 02 3F 00 88 01 00 8A 01 05 82 02 3F FF 8D 02 FF FF 84 06 A1
A2 A3 A4 A5 A6 8C 00 AB 00 9000

; ACOS6-SAM Card - load the bulk key(ID:82) to RAM
;diversify key ;no diversify
<SAM 80 72 05 02 00
>SAM 9000

;first authentication step 1
<Card 70 0B 40 00
>Card 90 <ek(RndB)>

<SAM 80 78 03 02 10 <ek(RndB)>
>SAM 6120
<SAM 00 C0 00 00 20
>SAM <ek(RndA + RndB')> 9000

;first authentication step 2
<Card 72 <ek(RndA + RndB')>
>Card 90 <TI><ek(RndA')><PICCcap2><PCDcap2>

;Verify the card return value
<SAM 80 7A 00 02 20 <TI><ek(RndA')><PICCcap2><PCDcap2>
>SAM 9000


;Read encrypted, no MAC on response, MAC on command
<SAM 80 74 A1 01 04 30 14 00 01
>SAM 610C
<SAM 00 C0 00 00 0C
>SAM 30 14 00 01 <8-byte MAC> 9000
```

```
<Card 30 14 00 01 <8-byte MAC>
<Card 90 <ek(16-byte data)>

;Process the response
<SAM 80 74 A3 01 10 <ek(16-byte data)>
>SAM 9000

; Decrypt the encrypted data
<SAM 80 76 A5 01 10<ek(16-byte data)>
>SAM 6110
<SAM 00 C0 00 00 10
>SAM <16-byte data> 9000


; Read encrypted, MAC on response, MAC on command
<SAM 80 74 A1 01 04 31 14 00 01
>SAM 610C
<SAM 00 C0 00 00 0C
>SAM 31 14 00 01 <8-byte MAC> 9000

<Card 31 14 00 01 <8-byte MAC>
<Card 90 <ek(16-byte data)><8-byte MAC>

;Process the response
<SAM 80 74 A3 01 10 <ek(16-byte data)>
>SAM 6108
<SAM 00 C0 00 00 08
>SAM <8-byte MAC> 9000

; Decrypt the encrypted data
<SAM 80 76 A5 01 10 <ek(16-byte data)>
>SAM 6110
<SAM 00 C0 00 00 10
>SAM <16-byte data> 9000


;Read encrypted, no MAC on response, no MAC on command
<SAM 80 74 A1 01 04 34 14 00 01
>SAM 6104
<SAM 00 C0 00 00 04
>SAM 34 14 00 01 9000

<Card 34 14 00 01
<Card 90 <ek(16-byte data)>

;Process the response
<SAM 80 74 A3 01 10 <ek(16-byte data)>
>SAM 9000

;Decrypt the encrypted data
<SAM 80 76 A5 01 10 <ek(16-byte data)>
>SAM 6110
<SAM 00 C0 00 00 10
>SAM <16-byte data> 9000


;Read encrypted, MAC on response, no MAC on command
<SAM 80 74 A1 01 04 35 14 00 01
>SAM 6104
<SAM 00 C0 00 00 04
>SAM 35 14 00 01 9000
```

```
<Card 35 14 00 01
<Card 90 <ek(16-byte data)><8-byte MAC>

;Process the response
<SAM 80 74 A3 01 10 <ek(16-byte data)>
>SAM 6108
<SAM 00 C0 00 00 08
>SAM <8-byte MAC> 9000

;Decrypt the encrypted data
<SAM 80 76 A5 01 10 <ek(16-byte data)>
>SAM 6110
<SAM 00 C0 00 00 10
>SAM <16-byte data> 9000
```

13.0.Sample Card Initialization

This section will provide more demonstrations on 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 as shown in the figure below. For the purpose of this demonstration, the script will not activate the card to user state and many files do not have the files activated. Therefore, do not treat this file system to be 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.

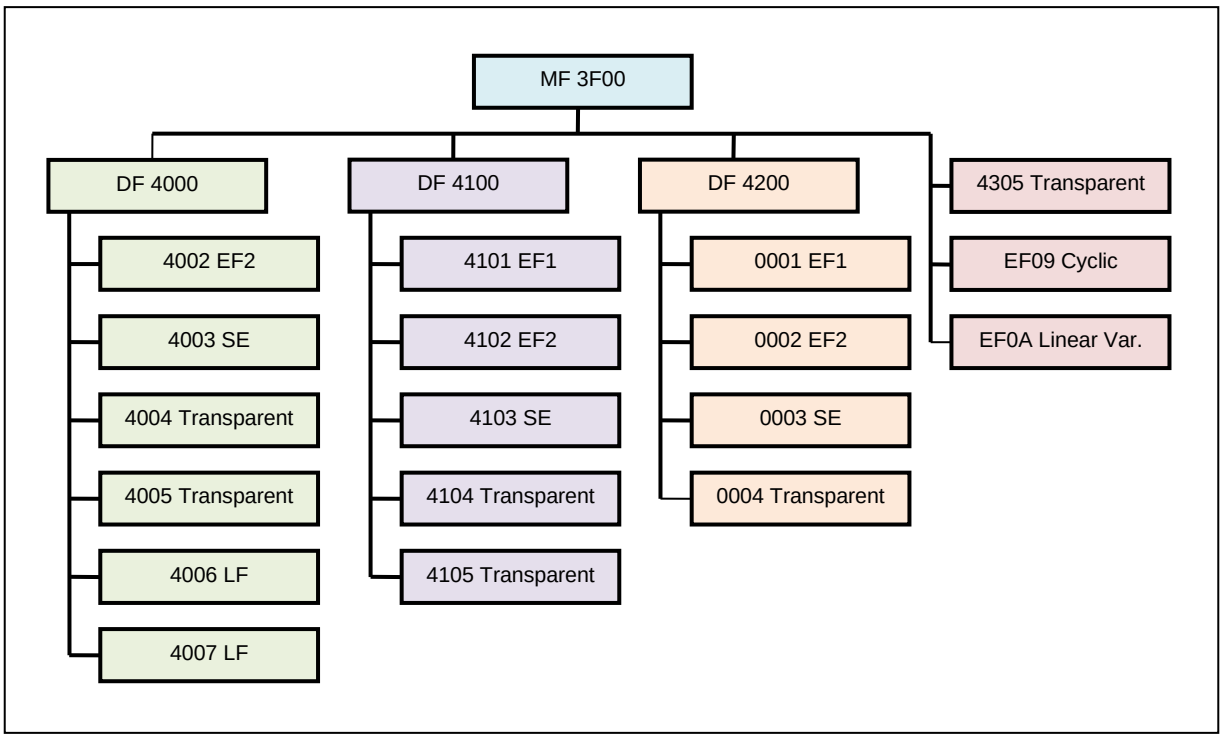


Figure 72: File System Example

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)

13.1. Personalization of Card Header

```
; Personalize Card ID Number
00 D6 EE C0 03 01 23 45 (9000)
; Set the Special Function Flags to allow Key Injection Only, Protect
Master Key and Deactivate Card Enable Flags

00 D6 EE F0 01 1F (9000)
```

13.2. 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
06 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 usages, 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 usages, 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 usages, 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 Variable 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)
```

13.3. 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)
```

Microsoft and Windows are registered trademarks of Microsoft Corporation in the United States and/or other countries.
MIFARE, MIFARE DESFire, MIFARE DESFire EV1, MIFARE Plus, MIFARE Ultralight, MIFARE Ultralight C are registered trademarks of NXP B.V. and are used under license.