TECHNICAL SPECIFICATIONS

BANKING CARDS

FC-PAY GP DDA – ST BLUE PEARL

PURPOSE

This document presents the full technical specifications (features, applications, personalization) of the FC-Pay GP DDA product based on ST BLUE PEARL operating system.

CONTENT

REFERENCES

[Ref 1] Sun Microsystems, JavaCard specifications, version 2.2.2
[Ref 2] GlobalPlatform, Card Specification, Version 2.1.1, March 2003
[Ref 3] EMVco, Integrated Circuit Card Specifications for Payment System 4.3, May 2004
[Ref 4] EMVco, Common Personalization Specifications, v1.0, June 2003
[Ref 5] Visa, GlobalPlatform 2.1.1 Card Implementation Requirements Version 1.0 May 2003
[Ref 6] Visa, ICC specifications, Version 1.5.4, March 2009 & March 2012
[Ref 7] VSDC personalization specification, Version 2.0, September 2009
[Ref 8] Visa, Technical Guide to Visa's applet for Global Platform Cards v1.3, May 2013
[Ref 9] MasterCard, M/Chip 4 Card Application Specifications for Credit and Debit Version 1.1
[Ref 10] MasterCard, M/Chip 4 Common Personalization Specifications, August 2003

MODIFICATIONS

Date	Version	Author	Detail
30/01/2017	1.0	YB	Document creation

1. PRODUCT FEATURES

FC-Pay GP DDA product is a state of the art JavaCard / Global platform smartcard compliant with the latest standards.

Main features:

- 8KB / 18KB / 40 Kbytes EEPROM versions available
- Crypto-processor (3DES, RSA algorithms)
- Java Card 2.2.2
- Global Platform 2.1.1
- EMV 4.3
- VSDC 1.5.4, PSE, PPSE, M/CHIP 4 applets in ROM
- Contact interface
- Common Personalization Specifications (CPS) compliant

Product is certified by international payment schemes Visa and MasterCard. Here are the details:

- VISA
  - Reference: LBUBIV2472
  - Date: June 21st, 2016
  - Validity: July 13th, 2027 (maximum validity of card on the field)
- MasterCard Worldwide
  - Letter of approval: CLOA-ENPI170103-170126(a)
  - CAST: CCN 2033
  - Validity: February 18th, 2019 (eligible for renewal)
  - Approval identifier: 03101A1600020000

2. TECHNICAL REFERENCE

2.1. OPERATING SYSTEM

This product was developed by ST Microelectronics / UbiVelox with the technical reference ST-PAY-J-BLUE-PEARL and also known as Ucard UBJ21-G24.

Card is using a **JavaCard / Global Platform** operating system therefore additional applets may be loaded in EEPROM. Here are the characteristics:

JavaCard 2.2.2 as per [Ref 1]

- Garbage collection

Global Platform 2.1.1 as per [Ref 2]

- Visa GP Configuration 3 as per [Ref 5]
- SCP02 with implementation option '15'
- Global PIN supported
- EMV Level 1 requirements

Memory resources:

- Persistent Java heap: 55 Kbytes
- Transaction buffer: 6 Kbytes
- Transient Java heap: 2.54 Kbytes
- APDU buffer: 261 bytes
- Java stack: 480 bytes
- Data block size in load command: 255 bytes

Security features: the following classes / fields are supported:

- RandomData
  - ALG_PSEUDO_RANDOM, ALG_SECURE_RANDOM
- Cipher:
  - ALG_DES_CBC: _ISO9797_M1, _ISO9797_M2, _NOPAD
  - ALG_DES_ECB: _ISO9797_M1, _ISO9797_M2, _NOPAD
  - ALG_RSA: _PKCS1, _NOPAD (maximum length 2048 bits)
  - MODE_DECRYPT, MODE_ENCRYPT
- Signature:
  - ALG_DES_MAC4: _ISO9797_M1, _ISO9797_M2, _NOPAD, _ISO9797_1_M2_ALG3
  - ALG_DES_MAC8: _ISO9797_M1, _ISO9797_M2, _NOPAD, _ISO9797_1_M2_ALG3
  - ALG_RSA: _SHA_ISO9796
  - MODE_SIGN, MODE_VERIFY
- Message digest:
  - SHA-1

- Key builder:
    - LENGTH_DES, LENGTH_DES3_2KEY, LENGTH_DES3_3KEY
    - TYPE_DES, TYPE_DES_TRANSIENT_DESELECT, TYPE_DES_TRANSIENT_RESET
    - LENGTH_RSA: multiple of 32 bits between 512 bits and 2048 bits
    - LENGTH_RSA_CRT_PRIME: multiple of 16 bits between 256 and 512 bits, and 1024 bits
    - TYPE_RSA_PUBLIC (with maximum key length of 2048 bits)
    - TYPE_RSA_PRIVATE (with maximum prime length of 2048 bits)
    - TYPE_RSA_CRT_PRIVATE (with maximum prime length of 1024 bits)
- Key generation:
    - RSA_KEYPAIR: key length up to 2048 bits (multiple of 32 bits)
    - RSA_KEYPAIR_CRT: key length up to 1024 bits

2.2. CHIP

Product uses ST Microelectronics **ST31H320** chip, with the following characteristics:

- CPU: Enhanced 8/16-bit ST23 CPU core with 16 Mbytes of linear addressable memory
- Memory:
    - ROM: 300 Kbytes
    - EEPROM: 40 Kbytes
    - RAM: 8 Kbytes
- EEPROM Write Operations:
    - 1 to 32 bytes erase/write operation in 1.0ms
    - Minimum of 500,000 write/erase cycles
    - Data retention for minimum 30 years
- Cryptography:
    - Enhanced NESCRYPT crypto-processor for public key cryptography
    - Three-key Triple DES accelerator (EDES+)
    - Three 8-bit timers with watchdog and interrupt capability
- Security
    - Active shield
    - Memory protection unit (MPU)
    - Monitoring of environmental parameters
    - Protection mechanisms against faults
    - AIS-31 class P2 compliant true random number generator (TRNG)
    - ISO 13239 CRC calculation block
    - Unique serial number on each die
- Clock Sources
    - External clock: up to 10 MHz
    - Internal clock: up to 28 MHz
- Operating conditions:
    - Voltage range: 1.62 V to 5.5 V supply voltage
    - Temperature range: -25°C to +85°C
    - Electrostatic discharge: over 5kV

2.3. INTERFACES

- Answer to Reset:
  - Default ATR: 3B680000 0073C84000009000
- Protocols:
  - ISO 7816 T=0
  - ISO 7816 T=1
- Baud rates:
  - From 9,600 bps (3.57 MHz) to 312,500 bps (5.00 MHz) - contact
- Logical channels: not supported

2.4. CPLC

Card production life cycle (CPLC) data contains the information regarding the development and production of the product. Here are the details:

Information	Data	Length	Value	Responsible
IC	Fabricator	2 bytes	4750	Chip manufacturer (STM)
	Type	2 bytes	00DE	
Operating system	Developer	2 bytes	5542	
	Release date	2 bytes	5293	
	Release level	2 bytes	0300	
IC manufacturing	Fabrication date	2 bytes	variable	
	Serial number	4 bytes	variable	
	Batch identifier	2 bytes	variable	
IC module manufacturing	Fabricator	2 bytes	5542	Module manufacturer (STM)
	Date	2 bytes	variable	
ICC manufacturing	Manufacturer	2 bytes	8453	Card manufacturer (FutureCard)
	Date	2 bytes	variable	
ICC pre-personalization	Pre-personalizer	2 bytes	8454	
	Date	2 bytes	variable	
	Equipment identifier	4 bytes	variable	
ICC personalization	Personalizer	2 bytes	variable	Card personalizer
	Date	2 bytes	variable	
	Equipment identifier	4 bytes	variable	

2.5. Applications

Following applets are available in ROM of the card:

- Issuer security domain
- Payment Systems Environment (PSE)
- Visa VSDC 1.5.4
- MasterCard M/Chip 4 Select

Please refer to the following chapter for more details.

2.5.1. Issuer security domain

ISD is supported in compliance with Global Platform (as per [Ref 2] and [Ref 5]).

Application Identifier (AID) is A000000151000000

Application supports the following **APDU commands**:

- DELETE
- INSTALL (for LOAD, INSTALL AND MAKE SELECTABLE, EXTRADITION)
- LOAD
- INITIALIZE UPDATE (SCP02, implementation option i = 15)
- EXTERNAL AUTHENTICATE (SCP02, implementation option i = 15)
- PUT KEY
- STORE DATA
- GET DATA
- GET STATUS, with P1 parameter:
  - ‘80’ (issuer security domain) supported
  - ‘40’ (applications and security domains) supported
  - ‘20’ (executable load file) supported
  - ‘10’ not supported
- SET STATUS

2.5.2. Payment System Environment (PSE)

Payment System Environment application allows to list the payment applications available in the card, and to use the PSE / directory selection method as per [Ref 3]. It supports standard CPS personalization ([Ref 4]).

Product uses the function included in VSDC applet (cf. 2.5.3)

Application Identifier (AID) for the application is: 315041592E5359532E4444463031.

Application supports the following **APDU commands**:

- SELECT
- READ RECORD

Personalization details are described in chapter 4.1.

2.5.3. Visa VSDC 1.5.4 application

Visa Smart Debit / Credit applet has been developed by Visa International and is known as '**VSDC 2.8.1g**' applet. It is compliant with [Ref 6], [Ref 7] and [Ref 8].

Application Identifier (AID) for the application is defined by the issuer according to the type of card. Main values are:

- A0000000031010 for Visa Debit / Credit
- A0000000032010 for Visa Electron

For the **contact application**, the following **features** from [Ref 6] are supported:

Transaction step	Details
Application selection	▪ FCI Issuer Discretionary Data (tag 'BF0C') supported ▪ Multiple VSDC applications (AIDs)
Initiate Application	▪ PDOL supported ▪ Geographic Restriction check
Offline Data Authentication	▪ Static Data Authentication (SDA) ▪ Dynamic Data Authentication (DDA) ▪ Combined DDA/AC (CDA)
Cardholder Verification	▪ Clear text offline PIN supported ▪ Enciphered PIN supported (using ICC public key) ▪ PIN Try counter retrievable via Get Data command
Terminal Action Analysis	▪ Terminal Velocity Checking supported
Card Action Analysis	▪ All velocity checking are supported ▪ Currency conversion supported ▪ All optional checks are supported
Online Processing	▪ Issuer Authentication (both mandatory and optional is supported) ▪ Cryptogram Version (CVN) 10 and 18 supported
Completion	▪ All velocity checking are supported ▪ Currency conversion supported ▪ All optional checks are supported
Issuer-to-Card Script Processing	▪ All issuer script commands are supported ▪ Cyclic Issuer Script Counter ▪ 4 and 8 byte MAC lengths supported
Personalization	▪ EMV Card Personalization Specification supported as per [Ref 4] and [Ref 7] – recommended
Other functions	▪ Multiple VSDC instances ▪ Application Block/Unblock Linking ▪ VSDC Shared PIN ▪ Issuer discretionary data (options 01, 02, 03, 04, 05, 06) ▪ Available offline spending amount

For more information, please refer to [Ref 8].

Note that this application may be used for **Dynamic Password Authentication (DPA)**.

Personalization details are described in chapter 4.2.

2.5.4. M/Chip 4 application

M/Chip Advance applet is compliant with [Ref 9] (v1.1b), and also supports CPS personalization as defined by EMVco and MasterCard ([Ref 10]).

It supports **M/Chip 4 Select (1.1b)** profile.

Application Identifier (AID) for the application is defined by the issuer according to the type of card. Main values are:

- A0000000041010 for MasterCard
- A0000000043060 for Maestro
- A0000000046000 for Cirrus

All the **features** defined in [Ref 9] are supported, for instance:

Transaction step	Details
Offline Data Authentication	▪ Static Data Authentication (SDA) ▪ Dynamic Data Authentication (DDA) ▪ Combined DDA/AC (CDA)
Cardholder Verification	▪ Clear text offline PIN supported ▪ Enciphered PIN supported ▪ PIN Try counter retrievable via Get Data command
Session key derivation	▪ MasterCard proprietary ▪ EMV CSK
Online processing	▪ EXTERNAL AUTHENTICATE not supported ▪ Issuer authentication included in 2nd GenerateAC
Issuer script	▪ Multiple issuer scripts supported ▪ No CARD BLOCK command
Personalization	▪ CPS personalization (as per [Ref 10])
Other functions	▪ Multiple M/Chip instances ▪ Possibility of sharing offline PIN between applications ▪ Transaction Logging

Note that this application may be used for **Cardholder Authentication Program (CAP)**.

Personalization details are described in chapter 4.3.

3. PERSONALIZATION

3.1. OVERVIEW

In order to facilitate the migration of card issuers, all FC-Pay products are compliant to the Common Personalization Specifications "CPS" (cf. [Ref 4]). This guarantees a much easier development, faster time to market, and no dependence on a specific card supplier.

This card being a Global Platform card, only the applet installation may be required before loading the data.

This chapter describes the different parts of the process required for personalization including the process (3.2), the APDU commands (3.3) and the cryptography (3.4).

The following chapter (4) describes the specific elements (installation and DGIs) of each application.

3.2. PERSONALIZATION PROCESS

Personalization process has 2 parts:

- Installation of the applications in the card – referred to as the "pre-personalization" in CPS
- Personalization of each application according to CPS

Note: there are several 'security levels' (00, 01 or 03) described in CPS, those options are described in the document, and it is the personalizer choice.

Note: the last application to be personalized is the CARD MANAGER (ISD), which will switch the card to SECURED state.

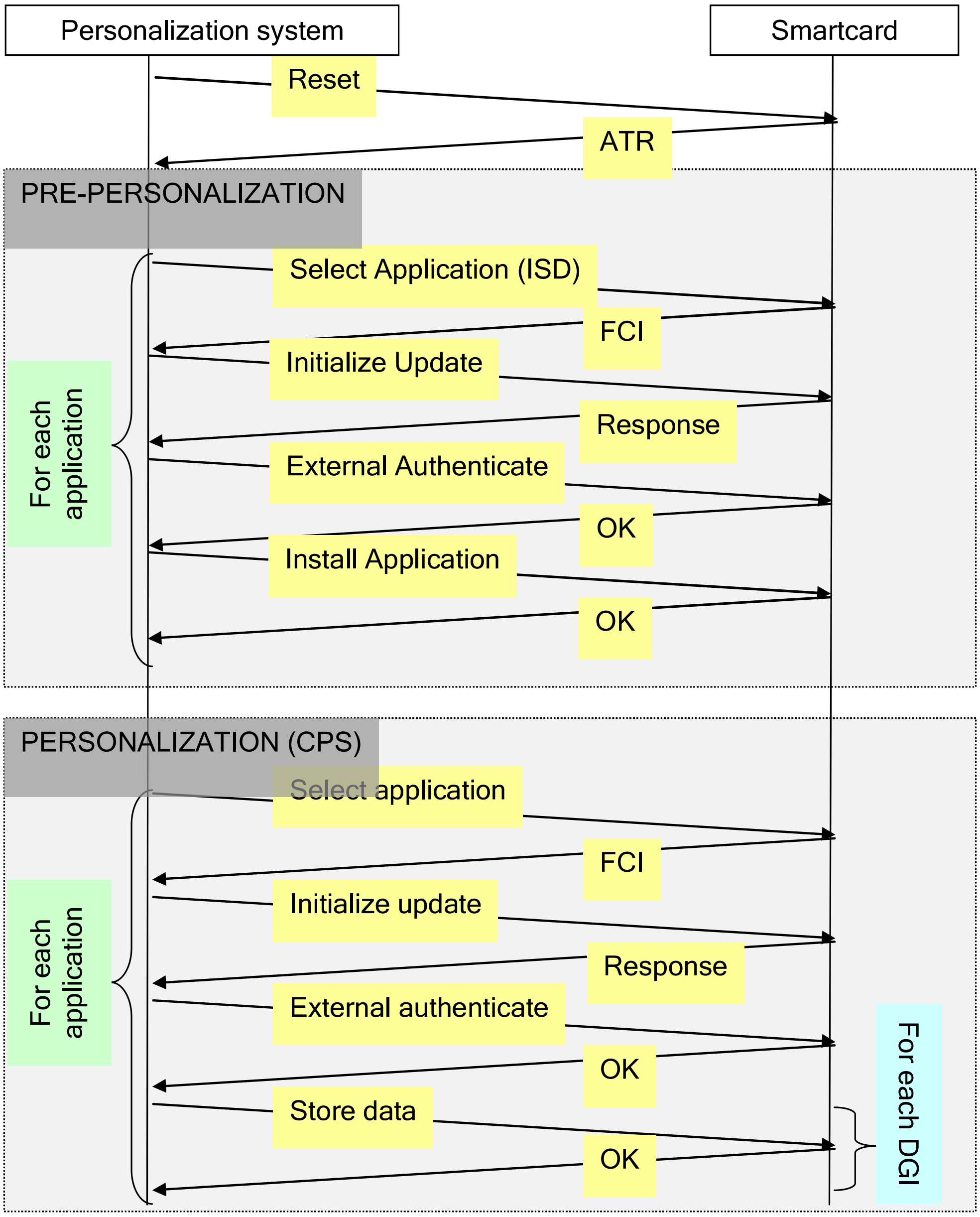
Note on the **card initial state**:

- As explained above, the general is that no application is created on the card before shipment to the card personalizer (in order to allow for full flexibility)
    - In some specific cases, application may be already installed upon request by the customer. Full technical details should be provided.
- Otherwise, the keys / authentication elements are as per CPS specification. **3 derived keys** are generated for each IC card and placed into the application. They are 16 bytes (112 bits plus parity) DES keys, derived from the KMC as detailed in chapter 3.4.1:
    - K_{ENC} is used to generate the card cryptogram and to verify the host cryptogram. This key is also used to decrypt the STORE DATA command data field in CBC mode if the security level of secure messaging requires the command data field to be encrypted. It is also used for the EXTERNAL AUTH PRO command.
    - K_{MAC} is used to verify the C-MAC for the EXTERNAL AUTHENTICATE command and also to verify the C-MAC for the STORE DATA command(s) if the security level of secure messaging requires a MAC of the command data.
    - K_{DEK} is used to decrypt in ECB mode secret data received in the STORE DATA command.

→ Note that for test purposes, FC-Pay cards use the following test KMC:

```
4755525557414C54455244534F555A41 (KCV: 4F2817)
```

Here is the **personalization commands flow**:



3.3. APDU COMMANDS

This chapter describes the APDU commands required for the personalization of the product. They are either:

- CPS commands (extract from [Ref 4] - for full reference, refer to the original document)
- Proprietary commands used to create the file system

All responses to commands, whether successfully processed or not, include two status bytes SW. The most common values are presented below.

3.3.1. SELECT

The SELECT command is used to select each IC card application to be personalized.

Reference

- [Ref 3]
- [Ref 4] – chapter 3.2.2

Command (Case 4)

CLA	INS	P1	P2	L_C
00	A4	04	‘00’: First or only occurrence ‘02’: Next occurrence	AID length

Data In

Field	Length	Presence	Value
AID	Variable (5 - 16 bytes)	Mandatory	Application Identifier

Data Out (TLV format)

Field	Presence	Tag	Length	Value
FCI template	Mandatory	‘6F’	variable	(below fields)
DF name	Mandatory	‘84’	variable (‘05’ - ’10)	AID value
FCI proprietary template	Mandatory	‘A5’	variable (may be ‘00’)	Data in TLV format (may be empty)

Status Words

SW	Description
‘9000’	Correct execution
‘6700’	Incorrect length
‘6A82’	The application does not exist

3.3.2. INITIALIZE UPDATE

The INITIALIZE UPDATE command is the first command issued to the IC card after the personalization device selects the application. INITIALIZE UPDATE is used to establish the Secure Channel Session to be used during personalization. The data to perform mutual authentication is exchanged. The identifier and version number for the KMC and the data to be used to derive the K_{ENC}, the K_{MAC} and the K_{DEK} for the application are also returned.

The INITIALIZE UPDATE command will be issued once for each IC card application to be personalized.

The Card Cryptogram returned should be verified before proceeding to the next step of personalization.

Reference

- [Ref 4] – chapter 3.2.3

Command (Case 4)

CLA	INS	P1	P2	L_C
80	50	00	00	08

Data In

Field	Length	Presence	Value
RTERM	8 bytes	Mandatory	Random number generated by the personalization system and used in host and card cryptogram computation

Data Out

Field	Presence	Length	Value
KEYDATA *(used in KMC derivation – cf. 3.4.1)*	Mandatory	10 bytes	Variable
KMC version number	Mandatory	1 byte	Variable
SCP protocol	Mandatory	1 byte	‘02’
Sequence counter	Mandatory	2 bytes	Variable
Card challenge (RCARD)	Mandatory	6 bytes	Variable
Card cryptogram	Mandatory	8 bytes	Computed (cf. 3.4.3)

Status Words

SW	Description
‘9000’	Correct execution
‘6700’	Incorrect length

3.3.3. EXTERNAL AUTHENTICATE

The EXTERNAL AUTHENTICATE command follows the INITIALIZE UPDATE command and is used to authenticate the personalization device to the IC card application. The EXTERNAL AUTHENTICATE command will be issued once for each application to be personalized.

Note on the **security level** (P1 parameter):

- It is up to the personalizer to define the security level:
  - ‘00’: No security - All subsequent commands received by the IC card application will not include any security (no C-MAC, no encryption of the entire command data string)
  - ‘01’: MAC - All subsequent commands received by the IC card application must contain a C-MAC
  - ‘03’: Encryption and MAC - subsequent commands received by the IC card application will include a C-MAC and the command data field will be encrypted by the SKU_{ENC}
- It applies to all commands following the EXTERNAL AUTHENTICATE command (not the EXTERNAL AUTHENTICATE itself).
- The encryption specified in the EXTERNAL AUTHENTICATE command does not impact the security specified by the value of P1 in the STORE DATA command (see chapter 3.3.4). If both the EXTERNAL AUTHENTICATE command and the STORE DATA command specify encryption, the data is encrypted twice.

Reference

- [Ref 4] – chapter 3.2.4

Command (Case 3)

CLA	INS	P1	P2	L_C
80	82	Security level (cf. above)	00	10

Data In

Field	Length	Presence	Value
Host cryptogram	8 bytes	Mandatory	Computed as detailed in 3.4.3
C-MAC	8 bytes	Mandatory	Computed as detailed in 3.4.4

Status Words

SW	Description
‘9000’	Correct execution
‘6700’	Incorrect length
‘6982’	C-MAC verification failed
‘6300’	Authentication of host cryptogram failed

3.3.4. STORE DATA

The STORE DATA command is used to personalize the EMV applications. It requires a secure channel to be established (successful EXTERNAL AUTHENTICATE command), and the format of the command depends on the chosen security level (‘00’, ‘01’ or ‘03’).

There will be one STORE DATA command for each data grouping (DGI) defined in each application (cf. chapter 4), in the order presented.

Reference

- [Ref 4] – chapter 3.2.5 and 3.2.6

Command (Case 3)

CLA	INS	P1	P2	L_C
‘80’ for security level ‘00’ ‘84’ otherwise	E2	Data indicator ▪ ‘00’ for data in clear format ▪ ‘60’ for encrypted data ▪ ‘80’ for last STORE DATA sent	Sequence number ▪ ‘00’ for first command ▪ Incremented for next commands (‘01’, ‘02’…)	Variable

Data In (TLV format)

Field	Length	Presence	Value
DGI	2 bytes	Mandatory	Variable
DGI length	Variable	Mandatory	Variable
DGI value	Variable	Mandatory	Variable If data requires encryption (P1 = ‘60’), it is encrypted as per 3.4.5)
C-MAC	8 bytes	Optional	Present if security level = ‘01’ or ‘03’ (computed as per 3.4.4)

Note: If security level = ‘03’, in addition to the C-MAC computation, the command data in field must be encrypted as described in 3.4.6 using SKU_{ENC}.

Status Words

SW	Description
‘9000’	Correct execution
‘6700’	Incorrect length
‘6A88’	Incorrect data
‘6A80’	Unknown DGI

3.3.5. GET DATA

Retrieves information from the chip – used to get the CPLC.

Reference

- [Ref 2] – chapter 9.3

Command (Case 2)

CLA	INS	P1	P2	L_C
80	CA	9F	7F	2D

Data Out

Field	Presence	Length	Value
CPLC – Tag	Mandatory	2 bytes	‘9F7F’
CPLC – Length	Mandatory	1 byte	‘2A’
CPLC – Value	Mandatory	Variable	▪ Chip Manufacturer (4 bytes) ▪ Operating system developer (6 bytes) ▪ Chip production (8 bytes) – bytes 3-6 used for KMC derivation (cf. 3.4.1) ▪ Module manufacturer (4 bytes) ▪ IC card manufacturer (4 bytes) ▪ IC Pre-personalizer (8 bytes) ▪ IC Personalizer (8 bytes)

Status Words

SW	Description
‘9000’	Correct execution
‘6700’	Incorrect length

3.3.6. INSTALL

The INSTALL command is used to create an instance of an application of an available applet (executable file).

Reference

- [Ref 2] – chapter 11.5

Command (Case 4)

CLA	INS	P1	P2	L_C
80	E6	‘0C’ Install and Make Selectable	00	08

Data In (LV format)

Field	Presence	Length	Value
Executable Load File AID	Mandatory	Variable (‘05’-‘10’)	Cf. detail for each application in chapter 4 5-16 bytes
Executable Module AID	Mandatory	Variable (‘05’-‘10’)	Cf. detail for each application in chapter 4 5-16 bytes
Application AID	Mandatory	Variable (‘05’-‘10’)	Cf. detail for each application in chapter 4 5-16 bytes
Privileges	Mandatory	‘01’	Cf. detail for each application in chapter 4 1 byte
Parameters	Mandatory	Variable	Cf. detail for each application in chapter 4 Variable length
Token Field	Mandatory	‘00’	Empty (not used)

Data Out

Field	Presence	Length	Value
Acknowledgement	Mandatory	1 bytes	‘00’

Status Words

SW	Description
‘9000’	Correct execution
’6581’	Memory failure
’6A80’	Incorrect parameters in data field
’6A84’	Not enough memory space
’6A88’	Referenced data not found

3.4. CRYPTOGRAPHY

This chapter details all the cryptographic elements required during the card personalization, as used in the APDU commands described previously:

- CPS processes:
  - Computation of card keys (KMC derivation)
  - Mutual authentication (session key derivation, host and card cryptogram computation)
  - C-MAC computation (for EXTERNAL AUTHENTICATE command, and optionally for STORE DATA commands depending security level)
  - DGI encryption (for the relevant DGI, as indicated in the STORE DATA command)
  - APDU encryption (optionally for STORE DATA commands depending security level)
  - KCV computation
- Proprietary computation of authentication cryptogram

3.4.1. Card keys computation

CPS compliant cards carry 3 card keys derived from a master key (KMC): K_{ENC}, K_{MAC} and K_{DEK} as specified in chapter 3.2. Here is how they are computed:

- K_{ENC} := 3DES-ECB(KMC) [Six least significant bytes of the KEYDATA || ’F0’ || ‘01’ || Six least significant bytes of the KEYDATA || ‘0F’ || ‘01’]

  Example: if KEYDATA=0000702801042820208D & KMC=4755525557414C54455244534F555A41

  K_{ENC} = 3DES-ECB(4755525557414C54455244534F555A41) [01042820208DF00101042820208D0F01]

  = C4C488F45FCFE133D120D4E81C002BC5

- K_{MAC} := 3DES-ECB(KMC) [Six least significant bytes of the KEYDATA || ’F0’ || ‘02’ || Six least significant bytes of the KEYDATA || ‘0F’ || ‘02’]

  Example: if KEYDATA=0000702801042820208D & KMC=4755525557414C54455244534F555A41

  K_{MAC} = 3DES-ECB(4755525557414C54455244534F555A41) [01042820208DF00201042820208D0F02]

  = 64458B39541BAD796F25EFA95855D2B9

- K_{DEK} := 3DES-ECB(KMC) [Six least significant bytes of the KEYDATA || ’F0’ || ‘03’ || Six least significant bytes of the KEYDATA || ‘0F’ || ‘03’]

  Example: if KEYDATA=0000702801042820208D & KMC=4755525557414C54455244534F555A41

  K_{DEK} = 3DES-ECB(4755525557414C54455244534F555A41) [01042820208DF00301042820208D0F03]

  = F462310AF8058EEE64B4A9DD5A9480B1

3.4.2. Session keys computation

Cf. [Ref 4], chapter 5.2

DES session keys are generated every time a secure channel is initiated. These session keys may be used for subsequent commands if secure messaging is required: SKU_{ENC}, SKU_{MAC}, and SKU_{DEK}.

- All encryption, decryption and MACing in commands that are sent to the IC card must be performed using session keys (SKU_{ENC}, SKU_{MAC}, and SKU_{DEK}).
- Session keys must be calculated using the triple DES algorithm (ECB mode, ISO 10116) and the base keys K_{ENC}, K_{MAC}, and K_{DEK} (cf. chapter 3.4.1) to produce SKU_{ENC}, SKU_{MAC}, and SKU_{DEK} respectively.
- The session keys must be calculated in CBC mode. Padding is not added prior to encryption. The 16 bytes of derivation data, when encrypted, will result in a 16-byte double length key.
  - SKU_{ENC} := 3DES-CBC(K_{ENC}) ['0182' || Sequence Counter || '000000000000000000000000']

    Example: if sequence counter = 0009 & K_{ENC} = C4C488F45FCFE133D120D4E81C002BC5

    SKU_{ENC} = 3DES-CBC(C4C488F45FCFE133D120D4E81C002BC5) [0182000900000000000000000000
    00000]
    = 0700CAABC7C8B8C73C78E2702748B83E

  - SKU_{MAC} := 3DES-CBC(K_{MAC}) ['0101' || Sequence Counter || '000000000000000000000000']

    Example: if sequence counter = 0009 & K_{MAC} = 64458B39541BAD796F25EFA95855D2B9

    SKU_{MAC} = 3DES-CBC(64458B39541BAD796F25EFA95855D2B9) [0101000900000000000000000000
    00000]

    = 6DDD89AC55FF785AE43CD1670B5D83AC

  - SKU_{DEK} := 3DES-CBC(K_{DEK}) ['0181' || Sequence Counter || '000000000000000000000000']

    Example: if sequence counter = 0009 & K_{DEK} = F462310AF8058EEE64B4A9DD5A9480B1

    SKU_{DEK} = 3DES-CBC(F462310AF8058EEE64B4A9DD5A9480B1) [0181000900000000000000000000
    00000]

    = 7A1A42EBA76BF8E65DCE80AE59289D04

The session keys must be calculated for each IC card application during processing of the INITIALIZE UPDATE command using a sequence counter provided by the IC card.

These session keys are used for all cryptography for personalizing the IC card application until the completion of the last STORE DATA command.

3.4.3. Mutual authentication: Host and Card cryptograms

During the IC personalization process (INITIALIZE UPDATE command and EXTERNAL AUTHENTICATE command) the IC card returns a MAC (the card cryptogram) and the personalization device sends a MAC (the host cryptogram) to the IC card. The IC card and the personalization device authenticate each other using these cryptograms.

- Input to the MAC is first padded to the right with ‘80’. The result is padded to the right with up to 7 bytes of ‘00’ to make the result 8 bytes long. This is defined in ISO/IEC 9797-1, as padding method 2.
- The full triple DES MAC is as defined in ISO 9797-1 as MAC Algorithm 1 with output transformation 1, without truncation, and with triple DES taking the place of the block cipher.
- All 64 bits of the final output block are used as the MAC created for personalization cryptograms.
- Verification of a cryptogram must be performed by computing a MAC based on the same parameters (and key) and then comparing the result with the cryptogram received.
- As a summary, here is how both cryptograms are computed:
  - Card cryptogram := MAC-3DES(SKU_{ENC}) [R_{TERM} (8 bytes) || Sequence Counter (2 bytes) || R_{CARD} (6 bytes) || ‘8000000000000000’]

    Example: If R_{TERM} = 0102030405060708, sequence counter = 0009, R_{CARD} = 43BE60D338C0 & SKU_{ENC} = 0700CAABC7C8B8C73C78E2702748B83E:

    Card cryptogram = MAC-3DES(0700CAABC7C8B8C73C78E2702748B83E) [0102030405060708000943BE60D338C080000000000000000]

    = AA4B224FFACF6269

  - Host cryptogram := MAC-3DES(SKU_{ENC}) [Sequence Counter (2 bytes) || R_{CARD} (6 bytes) || R_{TERM} (8 bytes) || ‘8000000000000000’]

    Example: If R_{TERM} = 0102030405060708, sequence counter = 0009, R_{CARD} = 43BE60D338C0 & SKU_{ENC} = 0700CAABC7C8B8C73C78E2702748B83E:

    Host cryptogram = MAC-3DES(0700CAABC7C8B8C73C78E2702748B83E) [000943BE60D338C00102030405060708800000000000000000]

    = 1B80EF5098EC2538

3.4.4. C-MAC

Secure messaging is required for EXTERNAL AUTHENTICATE command (see chapter 3.3.3) and STORE DATA command (see chapter 3.3.4) – if security level requires it.

- Commands using secure messaging must include an 8 byte C-MAC created by the personalization device and verified by the IC card prior to accepting the command. If the command C-MAC fails to verify successfully, the IC card must reject the command with SW1 SW2 = '6982' and close the secure channel.
- The C-MAC must be calculated as follows:
  - Concatenate the command header (CLA INS P1 P2 L_C) with the command data (excluding the C-MAC itself). The command header must be modified as follows:
    - The value of L_C in the data to compute the C-MAC must reflect the presence of the C-MAC in the command data, i.e. $L_C = L_C + 8$.
    - The class byte shall be modified to indicate that this APDU includes secure messaging. This is achieved by setting bit 3 of the class byte. For all the commands defined in this specification, the class byte of commands that contain C-MAC will be '84'. If both the STORE DATA command and the security level of the EXTERNAL AUTHENTICATE command specify encryption, the encryption required by the STORE DATA command will be done before the C-MAC is computed and the EXTERNAL AUTHENTICATE encryption will be done after the C-MAC is computed.
    - The specific rules are:
      - Data groupings that are sent to the IC card with a P1 setting in the STORE DATA command indicate that the data is encrypted under the SKU_{DEK} before the C-MAC is computed.
      - If the security level in the EXTERNAL AUTHENTICATE command indicates that both encryption and MACing are used, the C-MAC must be created on the original command data (includes data encrypted under SKU_{DEK}) then the APDU command data field is encrypted under SKU_{ENC}.
  - Prepend the C-MAC computed for the previous command and validated by the card to the left of the data requiring the MAC. The personalization device and the IC card must keep any C-MAC that has been validated by the IC card to use as the first block of data for a subsequent C-MAC generation.
  - Append a byte of '80' to the right of the data selected.
  - If the resultant data block length is a multiple of 8, no further padding is required. Otherwise, append up to 7 bytes of '00' until the length is a multiple of 8. Divide the block into 8-byte blocks with the leftmost 8 bytes (binary zeroes or C-MAC from previous command) being block 1.
  - An Initialization Vector (IV) of all zeros is always used.
  - C-MAC is computed as defined below, using SKU_{MAC} as the key.
- The process of generating a C-MAC is performed with single DES plus final triple DES MAC according to ISO 9797-1 as MAC Algorithm 3 with output transformation 3, without truncation, and with DES taking the place of the block cipher. This is also known as the "Retail MAC". Both the personalization device and the IC card must create the C-MAC. The IC card verifies the C-MAC by comparing the C-MAC it creates to the C-MAC in the command. Both the personalization device and the IC card must also save the verified C-MAC to be used as the first block in the next C-MAC creation or verification.
- As a summary, here is how the C-MAC are computed:

- C-MAC (EXTERNAL AUTHENTICATE) := MAC-DES(SKU_{MAC}) [‘8482’ || security level || ‘0010’ || Host cryptogram (8 bytes) || ‘80’]

Example: If Security level = 00, Host cryptogram = 1B80EF5098EC2538 & SKU_{MAC} = 6DDD89AC55FF785AE43CD1670B5D83AC:

C-MAC = MAC-DES (6DDD89AC55FF785AE43CD1670B5D83AC) [84820000101B80EF5098EC253880]

= 4F97A8CDBF9EFDCE

- C-MAC (STORE DATA) := MAC-DES(SKU_{MAC}) [Last C-MAC received (8 bytes) || ‘84E2’ || sequence number || L_C+8 (1 byte) || Data In (var.) || ‘80’ || padding with ’00…00’ if required]

Example: If last C-MAC = 9695AD1D70486644, Sequence number = 0001, Data In = 0101277025571254133390000001513D491260100000000000005F280200565F2009746573742063617264 (L_C=2A) & SKU_{MAC} = 6DDD89AC55FF785AE43CD1670B5D83AC:

C-MAC = MAC-DES (6DDD89AC55FF785AE43CD1670B5D83AC) [9695AD1D7048664484E200013201012770255712541333900000 1513D491260100000000000005F280200565F200974657374206361726480]

= 3EC69FEB9729DFE1

3.4.5. *DGI Encryption*

Loading of secure data elements such as keys and PIN require data encryption, as indicated in P1 parameter of STORE DATA command (cf. chapter 3.3.4).

- The data preparation function must encrypt DES and RSA keys and secret data e.g. PIN Block, with Triple DES in ECB mode using a Transport Key *(outside of the scope of this document)*
- The personalization device must encrypt keys and secret data with Triple DES in ECB mode using the session key $\mathrm{SKU}_{\mathrm{DEK}}$.
- Triple DES in ECB mode, as defined in ISO 10116, is used.
- In summary, here is the computation method:
  - For 3DES keys and PIN: EncryptedData := 3DES-ECB($\mathrm{SKU}_{\mathrm{DEK}}$) [ClearData]

```
Example: If ClearData = 104597E5A4A7A77308FB2F620480682094FB8AD6AEFD26F7FD767A5
         27929021C6143CEAED038AE73C7E352D945F7765D       &       SKUDEK       =
         7A1A42EBA76BF8E65DCE80AE59289D04:
EncryptedData = 3DES-ECB(7A1A42EBA76BF8E65DCE80AE59289D04) [104597E5A4A7A77308F
         B2F620480682094FB8AD6AEFD26F7FD767A527929021C6143CEAED038AE73C7E352D9
         45F7765D]

         = 29E20CC13F9156B10FE47FA4BCD4F5C4DD7A8D9C3AAC80CC118B4B80B4479A37265
         9FF8725C6CB18736097DB5C75BD0B
```

  - For RSA keys: EncryptedData := 3DES-ECB($\mathrm{SKU}_{\mathrm{DEK}}$) [ClearData || ‘80’ || padding with ’00…00’ if required]

```
Example: If ClearData = B8940CF653E0D59DA5EB369242F842EDF35F370B6D719AA77CB193D
         246AFA0EA87D45BF26465B09A2A37E42766E6C42C8C6174C7DD817D0610C5D92FBC2D
         8487 & SKUDEK = 7A1A42EBA76BF8E65DCE80AE59289D04:
EncryptedData = 3DES-ECB(7A1A42EBA76BF8E65DCE80AE59289D04) [B8940CF653E0D59DA5E
         B369242F842EDF35F370B6D719AA77CB193D246AFA0EA87D45BF26465B09A2A37E427
         66E6C42C8C6174C7DD817D0610C5D92FBC2D848780000000000000000]

         = 72125B242BC500BFB20F60E49B0F310CEDE9BB1CC6CFE3C2026270CBB0E448DC0B5
         3BF8104655BF903889B09163942674EF4C52883DFB10446AD8B7268DC0CC34A6BD003
         9D4EA954
```

3.4.6. APDU Encryption

If the security level set in EXTERNAL AUTHENTICATE command requires MAC and encryption, the personalization device must encrypt the APDU command data field in CBC mode using the session key SKU_{ENC} after the MAC has been computed.

- Input to the encryption process is first padded to the right with ‘80’. The result is padded to the right with up to 7 bytes of ‘00’ (possibly none) to make the input data a multiple of 8-byte blocks.
- Encryption of data must be done in Triple DES in CBC mode, as defined in ISO 10116 with an Initial Vector equal to '0000000000000000'.
- In summary, here is the computation method:
    - APDU encrypted data := 3DES-CBC(SKU_{ENC}) [APDU clear data || ‘80’ || padding with ’00…00’ if required]

Example: If clear APDU is 80E200001991041682027C00941008010100100105001801030120010100 & SKU_{DEK} = 7AA8DF1A37F4F41AFBC7E0579E768A45:

APDU encrypted data = 3DES-CBC(7AA8DF1A37F4F41AFBC7E0579E768A45) [91041682027C00941008010100100105001801030120010100800000000000000]

= 4B539D57B0812DAE8650C6112386F07E4C9DBBE22658E5DA23747144578475F7

→ APDU = 84E20000284B539D57B0812DAE8650C6112386F07E4C9DBBE22658E5DA23747144578475F7112EE5F0925432FE (last 8 bytes are the C-MAC computed as specified in chapter 3.4.4)

3.4.7. KCV computation

As part of pre-personalization or personalization processes, it might be required to compute Key Check Values (KCV) when loading 3DES keys in order to guarantee the integrity of the data.

- It is the 3 most significant bytes of the result of the 3DES encryption of a zero block by the key.
- In summary, here is the computation method:
  - KCV := LEFT(3DES (KEY) ['0000000000000000'] , 3)

```
Example: If KEY = 4755525557414C54455244534F555A41
KCV = LEFT (3DES(4755525557414C54455244534F555A41) [0000000000000000] , 3)
 = 4F2817
```

4. APPLICATIONS PERSONALIZATION

This chapter presents the data related to each application personalization:

- **Installation / pre-personalization**
  - Identifiers of Load File / Executable File / Instance
  - Privileges / Parameters

- **Personalization DGI**
  - Data for each DGI (Content description)
  - Encryption requirement
  - Data template to be used (if any)
  - Data format (TLV or plain)
  - Presence requirement:
    - M: mandatory
    - O: optional
    - C: conditional

4.1. Payment System Environment

Here is the detailed personalization information for the Payment System Environment (PSE) application.

4.1.1. Installation

Applet installation information:

- Load File AID: 05 315041592E
- Applet AID: 0E 315041592E5359532E4444463031
- Instance AID: 0E 315041592E5359532E4444463031
- Privileges: 01 00
- Parameters: 02 C900

Note that PSE applet is also available in VSDC applet 2.8.1f1 with load file AID A00000000316 and applet AID A0000000031650.

4.1.2. Personalization DGI

Here are the **DGI** supported by the PSE application.

DGI	Data content	Encrypt?	Template	Format	Requirement
01nn	Data for record nn ('01' – 'FF') in the directory elementary file of SFI 01	No	'70'	TLV	M
9102	File control information for PSE DDF: - Must include SFI of the directory elementary file (tag '88' - 1 byte) for PSE application - Additional data (5F2D, BF0C…)	No	'A5'	TLV	M

4.1.3. Log

Here is a sample installation / personalization log for a standard PSE application with 1 record:

Installation

```
(successful secure channel opening – case of security level = '00')
```

INSTALL 315041592E5359532E4444463031

```
* APDU: 80E60C002A || 05 315041592E || 0E 315041592E5359532E4444463031 || 0E315
        041592E5359532E4444463031 || 0100 || 02 C900 || 00
* RESP: 00
* SW12: 9000
```

Personalization

SELECT DF 315041592E5359532E4444463031

```
* APDU: 00A404000E || 315041592E5359532E4444463031
* RESP: 6F15 || 84 0E 315041592E5359532E4444463031 || A5 03 880101
* SW12: 9000
```

```
(successful secure channel opening – case of security level = '00')
```

STORE DATA 0101

```
* APDU: 80E20000 Lc || 0101 || Length (1 byte) || PSE record 1 (variable)
* SW12: 9000
```

STORE DATA 9102

```
* APDU: 80E28001 Lc || 9102 || FCI Length (1 byte) || FCI data (variable)
* SW12: 9000
```

END

4.2. VISA VSDC 1.5.4

Here is the detailed personalization information for the Visa VSDC 1.5.4 application.

4.2.1. Installation

Applet installation information

- Load File AID: 06 A00000000310
- Applet AID:
  - 07 A0000000031056 (standard VSDC)
  - 07 A000000003104D (multi-access)
- Instance AID: to be defined by the issuer, standard values are:
  - 07 A0000000031010 for Visa Debit / Credit
  - 07 A0000000032010 for Visa Electron
- Privileges: 01 10
- Parameters: 03 C90102 (Shared PIN option)

4.2.2. Personalization DGI

Here are the DGI supported by the VSDC / VCPS application (as per [Ref 7]).

DGI	Data content	Encrypt?	Template	Format	Requirement
8010	Offline PIN block (ISO 0, 1 or 2 format) – 8 bytes	Yes	No	Binary	C if offline PIN
9010	PIN related data ▪ PIN try counter (tag '9F17') – 1 byte ▪ PIN try limit - 1 byte	No	No	Binary	
8000	DES keys ▪ Unique Derivation Key (UDK) – 16 bytes ▪ Message Authentication DEA Key (MACK) – 16 bytes ▪ Data Encipherment DEA Key (ENCK) – 16 bytes	Yes	No	Binary	M
9000	DES key check values: ▪ UDK KCV – 3 bytes ▪ MACK KCV – 3 bytes ▪ ENCK KCV – 3 bytes	No	No	Binary	O
8201	ICC DDA private key – CRT constant C_a (q^{-1} mod p)	Yes	No	Binary	C if DDA or CDA
8202	ICC DDA private key – CRT constant C_{d2} (d_q = d mod q-1)	Yes	No	Binary	
8203	ICC DDA private key – CRT constant C_{d1} (d_p = d mod p-1)	Yes	No	Binary	
8204	ICC DDA private key – CRT constant C_q (q)	Yes	No	Binary	
8205	ICC DDA private key – CRT constant C_p (p)	Yes	No	Binary	
9102	File control information – contact	No	'A5'	TLV	M
9104	Response to GPO command ▪ Application Interchange Profile (tag '82') ▪ Application File Locator (tag '94')	No	No	TLV	M
xxnn	Record data for elementary file of SFI xx ('01' – '0A') and record number nn ('01' – 'FF') Refer to relevant specifications for recommended data elements location.	No	'70'	TLV	M

DGI	Data content	Encrypt?	Template	Format	Requirement
3000	Application common internal data ▪ ATC (tag '9F36') – to personalize non null value	No	No	TLV	O
3001	Card Internal Risk Management Data: ▪ Application Currency Code (tag '9F51') ▪ Application Default Action (tag '9F52') ▪ CTLI (tag '9F53') – or in DGI 3F57 ▪ CTTAL (tag '9F54') – or in DGI 3F58 ▪ Issuer Authentication Indicator (tag '9F56') ▪ Issuer Country Code (tag '9F57') ▪ LCOL (tag '9F58') – or in DGI 3F56 ▪ UCOL (tag '9F59') – or in DGI 3F56 ▪ CTTAUL (tag '9F5C') – or in DGI 3F58 ▪ AOSA ('9F5D') – 1 byte value ▪ CTIUL (tag '9F5E') – or in DGI 3F57 ▪ CTICL (tag '9F72') – or in DGI 3F57 ▪ Currency Conversion Parameters (tag '9F73')	No	No	TLV	M Each TLV presence depends on function activation
3F56	Counters Data Template ▪ CTC 1 (tag 'DF11') ▪ CTCL 1 (tag 'DF21') – replaces '9F58' in DGI 3001 ▪ CTCUL 1 (tag 'DF31') – replaces '9F59' in DGI 3001	No	No	TLV	O can replace data elements from DGI 3001
3F57	International Counters Data ▪ CTCI 1 (tag 'DF11') ▪ CTCIL 1 (tag 'DF21') – replaces '9F53' in DGI 3001 ▪ CTCIUL 1 (tag 'DF31') – replaces '9F5E' in DGI 3001 ▪ CTCIC 1 (tag 'DF51') ▪ CTCICL 1 (tag 'DF61') – replaces '9F72' in DGI 3001	No	No	TLV	
3F58	Amounts Data Template ▪ CTTA 1 (tag 'DF11') ▪ CTTAL 1 (tag 'DF2x') – replaces '9F54' in DGI 3001 ▪ CTTAUL 1 (tag 'DF31') – replaces '9F5C' in DGI 3001	No	No	TLV	
3F5B	Application Internal Data Template ▪ Application Capabilities (tag 'DF01')	No	No	TLV	O
9200	Issuer application data (tag '9F10') IDD formats '02', '03', '04', '05' and '06' supported	No	No	TLV	M

- DGI containing the AFL (9104) should be sent before all SFI data (DGI xxnn), as content of AFL is used to initialize the SFI records.
- Enciphered PIN is supported with the DDA ICC key

4.2.3. Log

Here is a sample personalization log for a standard VISA VSDC 1.5.4 DDA profile with Visa Debit/Credit AID, 2 records in SFI1, 3 records in SFI2, 2 records in SFI3.

Installation

```
(successful secure channel opening – case of security level = '00')
```

INSTALL A0000000031010

```
* APDU: 80E60C001E || 06 A00000000310 || 07 A0000000031056 || 07 A0000000031010
        || 01 10 || 03 C90102 || 00
* RESP: 00
* SW12: 9000
```

Personalization

SELECT DF A0000000031010

```
* APDU: 00A4040007 || A0000000031010
* RESP: 6F 0B || 84 07 A0000000031010 || A5 00
* SW12: 9000
```

```
(successful secure channel opening – case of security level = '00')
```

STORE DATA 9102

```
* APDU: 80E20000 Lc || 9102 || Length (1 byte) || FCI data – contact (variable)
* SW12: 9000
```

STORE DATA 9104

```
* APDU: 80E20001 Lc || 9104 || Length (1 byte) || GPO response data – contact
        (variable)
* SW12: 9000
```

STORE DATA 0101

```
* APDU: 80E20002 Lc || 0101 || Length (1 byte) || SFI 1 – record 1 data
        (variable)
* SW12: 9000
```

STORE DATA 0102

```
* APDU: 80E20003 Lc || 0102 || Length (1 byte) || SFI 1 – record 2 data
        (variable)
* SW12: 9000
```

STORE DATA 0201

```
* APDU: 80E20004 Lc || 0201 || Length (1 byte) || SFI 2 – record 1 data
        (variable)
* SW12: 9000
```

STORE DATA 0202

```
* APDU: 80E20005 Lc || 0202 || Length (1 byte) || SFI 2 – record 2 data
        (variable)
* SW12: 9000
```

STORE DATA 0203

```
* APDU: 80E20006 Lc || 0203 || Length (1 byte) || SFI 2 – record 3 data
        (variable)
* SW12: 9000
```

STORE DATA 0301

```
* APDU: 80E20007 Lc || 0301 || Length (1 byte) || SFI 3 – record 1 data
        (variable)
* SW12: 9000
```

STORE DATA 0302

```
* APDU: 80E20008  Lc || 0302 ||  Length  (1  byte) || SFI  3  –  record  2  data
        (variable)
* SW12: 9000
```

STORE DATA 3001

```
* APDU: 80E20009 Lc || 3001 || Length (1 byte) || CRM data (variable)
* SW12: 9000
```

STORE DATA 9200

```
* APDU: 80E2000A Lc || 9200  || Length (1 byte)  || Issuer  application  data
        (variable)
* SW12: 9000
```

STORE DATA 8201

```
* APDU: 80E2600B Lc || 8201 || Length (1 byte) || Encrypted ICC DDA private key
        – Ca (variable)
* SW12: 9000
```

STORE DATA 8202

```
* APDU: 80E2600C Lc || 8202 || Length (1 byte) || Encrypted ICC DDA private key
        – Cd2 (variable)
* SW12: 9000
```

STORE DATA 8203

```
* APDU: 80E2600D Lc || 8203 || Length (1 byte) || Encrypted ICC DDA private key
        – Cd1 (variable)
* SW12: 9000
```

STORE DATA 8204

```
* APDU: 80E2600E Lc || 8204 || Length (1 byte) || Encrypted ICC DDA private key
        – Cq (variable)
* SW12: 9000
```

STORE DATA 8205

```
* APDU: 80E2600F Lc || 8205 || Length (1 byte) || Encrypted ICC DDA private key
        – Cp (variable)
* SW12: 9000
```

STORE DATA 8000

```
* APDU: 80E2601033 || 8000 || 30 || Encrypted MK_AC (16 bytes) || encrypted MK_SMI
        (16 bytes) || encrypted MK_SMC (16 bytes)
* SW12: 9000
```

STORE DATA 9000

```
* APDU: 80E2001133 || 9000 || 09 || MK_AC  KCV  (3  bytes)  || MK_SMI  KCV  (3  bytes)
        || MK_SMC KCV (9 bytes)
* SW12: 9000
```

STORE DATA 8010

```
* APDU: 80E260120B || 8010 || 08 || encrypted PIN block (8 bytes)
* SW12: 9000
```

STORE DATA 9010

```
* APDU: 80E2801305 || 9010 || 02 || PIN try counter(1 byte) || PIN try limit (1
        byte)
* SW12: 9000
```

END

4.3. MasterCard M/Chip 4

Here is the detailed personalization information for the M/Chip 4 application.

4.3.1. Installation

Applet installation information:

- Load File AID: 06 A00000000410
- Applet AID: 07 A0000000041010
- Instance AID: to be defined by the issuer, standard values are:
  - 07 A0000000041010 for MasterCard
  - 07 A0000000043060 for Maestro
  - 07 A0000000046000 for Cirrus
- Privileges: 01 00
- Parameters:
  - 04 C90210C0 (Shared PIN option, M/Chip 4 v1.1a)
  - 04 C9021080 (Shared PIN option, M/Chip 4 v1.1b)

4.3.2. Personalization DGI

Here are the DGI supported by the M/CHIP application (as per [Ref 10]).

DGI	Data content	Encrypt?	Template	Format	Requirement
xxnn	Record data for elementary file of SFI xx (‘01’ – ‘0A’) and record number nn (‘01’ – ‘FF’)	No	‘70’	TLV	O
A001	File Control Information (FCI) ▪ DF name (tag ‘84’) ▪ Proprietary template (tag ‘A5’)	No	‘6F’	TLV	M
A002	CRM data – 75 bytes ▪ Application Control (tag ‘D5’) – 2 bytes ▪ Default ARPC Response Code (tag ‘D6’) - 2 bytes ▪ Lower Consecutive Offline Limit (tag ‘9F14’) - 1 byte ▪ Upper Consecutive Offline Limit (tag ‘9F23’) - 1 byte ▪ Lower Cumulative Offline Transaction Amount (tag ‘CA’) - 6 bytes ▪ Upper Cumulative Offline Transaction Amount (tag ‘CB’) - 6 bytes ▪ Card Issuer Action Code – Decline (tag ‘C3’) - 3 bytes ▪ Card Issuer Action Code – Default (tag ‘C4’) - 3 bytes ▪ Card Issuer Action Code – Online (tag ‘C5’) - 3 bytes ▪ CRM Currency Code (tag ‘C9’) - 2 bytes ▪ Currency Conversion Table (tag ‘D1’) - 25 bytes ▪ CRM Country Code (tag ‘C8’) - 2 bytes ▪ CDOL 1 Related Data Length (tag ‘C7’) - 1 byte ▪ Additional Check Table (tag ‘D3’) - 18 bytes	No	No	Binary	M
A005	GPO response ▪ Application Interchange Profile (tag ‘82’) ▪ Application File Locator (tag ‘94’)	No	No	Binary	M

DGI	Data content	Encrypt?	Template	Format	Requirement
A006	IDN key – 16 bytes	Yes	No	Binary	M
A007	CRM data – 8 bytes Case of M/CHIP 4 v1.1a ▪ Application Transaction Counter Limit – 2 bytes ▪ Previous Transaction History – 1 byte ▪ MAC in Script Counter Limit - 1 byte ▪ Global MAC in Script Counter Limit - 3 bytes ▪ Key Derivation Index – 1 byte Case of M/CHIP 4 v1.1b ▪ Application Transaction Counter Limit – 2 bytes ▪ Previous Transaction History – 1 byte ▪ AC Session Key Counter Limit – 2 bytes ▪ SMI Session Key Counter Limit – 2 bytes ▪ Key Derivation Index – 1 byte	No	No	Binary	M
A008	Bad cryptogram counter limit – 2 bytes	No	No	Binary	M
A009	Application life cycle data – 48 bytes First 8 bytes should contain the approval identifier (cf. 1)	No	No	Binary	M
8000	DES keys (contact) ▪ AC master key (MK_{AC}) – 16 bytes ▪ SM for integrity master key (MK_{SMI}) – 16 bytes ▪ SM for confidentiality master key (MK_{SMC}) – 16 bytes	Yes	No	Binary	M
9000	DES key check values: ▪ MK_{AC} KCV – 3 bytes ▪ MK_{SMI} KCV – 3 bytes ▪ MK_{SMC} KCV – 3 bytes	No	No	Binary	O
8201	ICC DDA private key – CRT constant C_a (q^{-1} mod p)	Yes	No	Binary	M
8202	ICC DDA private key – CRT constant C_{d1} (dq = d mod q-1)	Yes	No	Binary	
8203	ICC DDA private key – CRT constant C_{d2} (dp = d mod p-1)	Yes	No	Binary	
8204	ICC DDA private key – CRT constant C_q (q)	Yes	No	Binary	
8205	ICC DDA private key – CRT constant C_p (p)	Yes	No	Binary	
8301	ICC PIN Encipherment private key – CRT constant C_a (q^{-1} mod p)	Yes	No	Binary	C if dedicated key for encrypted offline PIN
8302	ICC PIN Encipherment private key – CRT constant C_{d1} (dq = d mod q-1)	Yes	No	Binary	
8303	ICC PIN Encipherment private key – CRT constant C_{d2} (dp = d mod p-1)	Yes	No	Binary	
8304	ICC PIN Encipherment private key – CRT constant C_q (q)	Yes	No	Binary	
8305	ICC PIN Encipherment private key – CRT constant C_p (p)	Yes	No	Binary	
8010	Offline PIN block (ISO 2 format) – 8 bytes	Yes	No	Binary	C if offline PIN
9010	PIN related data ▪ PIN try counter (9F17) – 1 byte ▪ PIN try limit – 1 byte	No	No	Binary	

4.3.3. *Log*

Here is a sample personalization log for a standard M/Chip 4 Select with MasterCard AID, 1 record in SFI 1, 3 records in SFI 2 and 2 records in SFI 3.

Installation

```
(successful secure channel opening – case of security level = '00')
```

INSTALL A0000000041010

```
* APDU: 80E60C001F || 06 A00000000410 || 07 A0000000041010 || 07 A0000000041010
        || 01 00 || 04 C9021080 || 00
* RESP: 00
* SW12: 9000
```

Personalization

SELECT DF A0000000041010

```
* APDU: 00A4040007 || A0000000041010
* RESP: 6F 17 || 8407A0000000041010 || A5 0C 500A4D415354455243415244 (FCI data
        )
* SW12: 9000
```

```
(successful secure channel opening – case of security level = '00')
```

STORE DATA 0101

```
* APDU: 80E20000 Lc || 0101 || Length (1 byte) || SFI 1 – record 1 data
        (variable)
* SW12: 9000
```

STORE DATA 0201

```
* APDU: 80E20001 Lc || 0201 || Length (1 byte) || SFI 2 – record 1 data
        (variable)
* SW12: 9000
```

STORE DATA 0202

```
* APDU: 80E20002 Lc || 0202 || Length (1 byte) || SFI 2 – record 2 data
        (variable)
* SW12: 9000
```

STORE DATA 0203

```
* APDU: 80E20003 Lc || 0203 || Length (1 byte) || SFI 2 – record 3 data
        (variable)
* SW12: 9000
```

STORE DATA 0301

```
* APDU: 80E20004 Lc || 0301 || Length (1 byte) || SFI 3 – record 1 data
        (variable)
* SW12: 9000
```

STORE DATA 0302

```
* APDU: 80E20005 Lc || 0302 || Length (1 byte) || SFI 3 – record 2 data
        (variable)
* SW12: 9000
```

STORE DATA A002

```
* APDU: 80E200064E || A002 || 4B || CRM data (variable)
* SW12: 9000
```

STORE DATA A005

```
* APDU: 80E20007 Lc || A005 || Length (1 byte) || AIP (2 bytes) || AFL (variabl
        e)
* SW12: 9000
```

STORE DATA A006

```
* APDU: 80E2600813 || A006 || 10 || Encrypted MK$_{IDN}$ (16 bytes)
* SW12: 9000
```

STORE DATA A007

```
* APDU: 80E200090B || A007 || 08 || CRM data (variable)
* SW12: 9000
```

STORE DATA A008

```
* APDU: 80E2000A05 || A008 || 02 || CRM data (variable)
* SW12: 9000
```

STORE DATA A009

```
* APDU: 80E2000B33 || A009 || 30 || ALCD data (variable)
* SW12: 9000
```

STORE DATA 8000

```
* APDU: 80E2600C33 || 8000 || 30 || Encrypted MK$_{AC}$ (16 bytes) || encrypted MK$_{SMI}$
        (16 bytes) || encrypted MK$_{SMC}$ (16 bytes)
* SW12: 9000
```

STORE DATA 9000

```
* APDU: 80E2000D0C || 9000 || 09 || MK$_{AC}$ KCV (3 bytes) || MK$_{SMI}$ KCV (3 bytes) ||
         MK$_{SMC}$ KCV (3 bytes)
* SW12: 9000
```

STORE DATA 8201

```
* APDU: 80E2600E Lc || 8201 || Length (1 byte) || Encrypted ICC DDA private key
        – Ca (variable)
* SW12: 9000
```

STORE DATA 8202

```
* APDU: 80E2600F Lc || 8202 || Length (1 byte) || Encrypted ICC DDA private key
        – Cd1 (variable)
* SW12: 9000
```

STORE DATA 8203

```
* APDU: 80E26010 Lc || 8203 || Length (1 byte) || Encrypted ICC DDA private key
        – Cd2 (variable)
* SW12: 9000
```

STORE DATA 8204

```
* APDU: 80E26011 Lc || 8204 || Length (1 byte) || Encrypted ICC DDA private key
        – Cp (variable)
* SW12: 9000
```

STORE DATA 8205

```
* APDU: 80E26012 Lc || 8205 || Length (1 byte) || Encrypted ICC DDA private key
        – Cq (variable)
* SW12: 9000
```

STORE DATA 8010

```
* APDU: 80E260130B || 8010 || 08 || encrypted PIN block (8 bytes)
* SW12: 9000
```

STORE DATA 9010

```
* APDU: 80E2001405 || 9010 || 02 || PIN try counter (1 byte) || PIN try limit
        (1 byte)
* SW12: 9000
```

STORE DATA A001

```
* APDU: 80E28015  Lc  || A001 || Length (1 byte) || FCI (variable, template 6F)
* SW12: 9000
```

END

4.4. Card Manager

As a mandatory last step of the personalization process, the Card Manager needs to be personalized in order to secure the card.

4.4.1. Installation

No **installation or pre-personalization** required.

Card Manager is already present (application identifier of the Issuer Security Domain to be used - cf. 2.5.1).

4.4.2. Personalization DGI

Here are the **DGI** supported by the Card Manager application (standard EMV DGI and specific ones).

DGI	Data content	Encrypt?	Template	Format	Requirement
9F66	CPLC – personalization – 8 bytes ▪ Personalizer – 2 bytes ▪ Personalization date – 2 bytes (YDDD format) ▪ Personalization equipment identifier – 4 bytes	No	No	V	M
9F70	End of personalization – 1 byte (value: ‘0F’)	No	No	V	M

4.4.3. Log

Here is a sample installation / personalization log:

Installation

N/A

Personalization

SELECT DF A000000003000000

```
* APDU: 00A4040008 || A000000151000000
* RESP: 6F10 || 84 08 A000000151000000|| A5 04 9F6501FF
* SW12: 9000
```

```
(successful secure channel opening - case of security level = ‘00’)
```

STORE DATA 9F66

```
* APDU: 80E200000B || 9F66 || 08 || CPLC-personalization data (8 bytes)
* SW12: 9000
```

STORE DATA 9F70

```
* APDU: 80E2800104 || 9F70 || 01 || 0F
* SW12: 9000
```

END

END OF DOCUMENT