IoT Security Applet Interface Description

gsma.com/IoTSecurity

IoT Security Applet Interface Description

Version 1.0

03/12/2019

This is a Non-binding Permanent Reference Document of the GSMA

Security Classification: Non-confidential

Copyright Notice

Disclaimer

Antitrust Notice

Table of Contents

1 Introduction

1.1 Overview

This is a companion document to GSMA document IoT.04 “Common Implementation Guide to Using the SIM as a ‘Root of Trust’ to Secure IoT Applications” [1].

Please read the GSMA document IoT.04 [1] to obtain all background information before reading this document.

1.2 Scope

This document defines the IoT device middleware to IoT security applet interface for:

- IoT Security Applet Type 1
- IoT Security Applet Type 2

as defined within section 3.2.3 of the GSMA document IoT.04 “Common Implementation Guide to Using the SIM as a ‘Root of Trust’ to Secure IoT Applications” [1].

1.3 Definitions

Term	Description
eUICC	A removable or non-removable UICC which enables the remote and/or local management of Profiles in a secure way. NOTE: The term originates from "embedded UICC".
UICC	A secure element platform specified in ETSI TS 102 221 [2].

1.4 Abbreviations

Abbreviation	Description
APDU	Application Protocol Data Unit
API	Application Programming Interface
DTLS	Datagram Transport Layer Security
ECC	Elliptic Curve Cryptography
ECDH	Elliptic Curve Diffie-Hellman
ECDHE	Elliptic Curve Diffie-Hellman Ephemeral
ECDSA	Elliptic Curve Digital Signature Algorithm
ECKA	Elliptic Curve Key Agreement
EF	Elementary File
GSM	Global System for Mobile
GSMA	GSM Association
HKDF	Hash-based Key Derivation Function
HMAC	Hash-based Message Authentication Code
ID	Identity
INS	Instruction
IKM	Input Key Material

Abbreviation	Description
IoT	Internet of Things
NIST	National Institute for Science and Technology
PRF	Pseudorandom Function
PSK	Pre-Shared Key
RFU	Reserved for Future Use
RSA	Rivest / Shamir / Adleman
SHA	Secure Hash Algorithm
TLS	Transport Layer Security
TLV	Type Length Value

1.5 References

Ref	Doc Number	Title
[1]	GSMA IoT.04	Common Implementation Guide to Using the SIM as a 'Root of Trust' to Secure IoT Applications. <LINK>
[2]	ETSI TS 102 221	Smart Cards; UICC-Terminal interface; Physical and logical characteristics (Release 16) <LINK>

2 Device-to-Applet Interface for IoT Security Applet Type 1

2.1 Summary of IoT Security Applet Type 1

The functional requirements for IoT Security Applet Type 1 are defined in document GSMA IoT.04 [1].

The applet is designed as a generic crypto toolbox application with specific (D)TLS (v1.2 and v1.3) related services.

The applet is based on a store holding objects, dedicated services to discover the capabilities of these objects, and services using these objects.

The services exposed to the device are those listed in section 2.3.

In this concept, the IoT security server (or the e/UICC issuer at personalization stage) is the one initializing the applet store: it creates necessary containers, sets object attributes, and when relevant, sets objects value. After this commissioning phase, the device can then request operations on objects present in the store but in this version it cannot create any objects, or reconfigure them.

IoT Security Applet Type 1 supports services based on asymmetric cryptography and symmetric cryptography. Those related to symmetric cryptography and RSA are optional.

2.2 Overview of the Applet Store

The Applet Store has the following properties:

- It contains objects that are public keys, private keys, secret keys and files.

- An optional label (1 to 60 bytes) and an identifier (1 to 20 bytes) are given to each object (public key, private key, secret key and file). They are used as an object reference in various services.
- Labels are associated to object containers while identifiers may be associated to containers or to the content. For instance, public key identifiers can be used as key check value.

  Note: It is assumed that label & ID policies set on the IoT security server are known on the device side. The device knows if IDs are used to identify containers or content and can thus use labels or IDs accordingly.

- Inside a type of object (public key, private key, secret key, files) object reference (identifiers, labels) must be unique. This rule does not apply to objects of different type.

Keys in the Applet Store have these attributes:

- Object access conditions: Specifies if the object (key) can be read or written from the device API.
- Object state: Specifies if the container holds an object (key, file) ready to use (by opposition to an empty or partially loaded key).
- Key type: Specifies what ECC named curve is supported, what RSA key length is supported.
- Key specific usage [asymmetric keys only]: keys can be tagged for specific usage like TLS handshake Certificate Verify message signature generation.
- Cryptographic functions: Specifies what cryptographic operation is allowed with the key. For example signature, key generation, key agreement.
- Algorithms for signature [asymmetric keys only]: Specifies what signature algorithms are usable with the key.
- Algorithms for hash: Specifies what hash algorithm is allowed. This applies to signature algorithms and key derivation where different hashes are accepted.
- Algorithms for key agreement [asymmetric keys only]: Specifies what key agreement schemes are usable with the key.
- Algorithms for key derivation [secret keys only]: Specifies what key derivation schemes are usable with the key.

Files in the Applet Store have these attributes:

- Object access conditions: Specifies if the object (file) can be read or written from the device API.
- Object state: Specifies if the container holds an object (file) ready to use (in contrast to an empty or partially loaded file).
- File specific usage: files can be tagged for specific usage like x509v3 certificate storage.

  Note: The applet may, as an implementer choice, leverage ‘specific usages’ attributes to implement checks related to the data being processed. The definition of these checks is out of scope.

Note: Private keys shall never leave the applet store. For that reason the read access condition is never set for private keys.

The store contains single asymmetric keys (only the private or only the public). Should single keys be associated to form a key pair, they must have the same label. Note: This constraint does not apply to key IDs.

In this situation (public key and private key forming a key pair) all attributes except key object state must have the same value.

It is assumed that the store might be updated by a remote administration server who can add, delete, or update objects from the applet store:

- In a situation where the execution of an operation requested by the remote administration server on a given applet store object is inconsistent with the possibility to execute a service requested by the device, this last request is denied, and the service requested by the device is acknowledged with a dedicated SW advising that a remote administration operation is on-going.
- Upon remote administration operation completion, the next service requested by the device is executed normally but specific SW is applied to inform the device that the applet store has changed.
- Note: In this situation, a device should refresh its knowledge of the applet store using "*Get Data"* services.

2.3 Applet Services

The applet provides the following services:

- Discover what feature the applet can support.
- Discover what objects are there in the applet store and their attributes.

2.3.1 Applet Services to Run a (D)TLS1.2 Handshake

For PKI schemes the applet provides:

- Get a random.
- Put a public key in the applet store (issuer certificate deployment model).
- Verify a signature.
- Generate a signature.
- Generate an ECC key pair.
- Compute the Diffie-Hellman shared secret (pre-master secret).
- (optional) Compute the master secret with the PRF function.

For the PSK-Plain scheme the applet provides:

- Get a random.
- Retrieve the PSK-Identity.
- Compute the master secret with the PRF function

For Pthe SK-ECDHE scheme the applet provides:

- Get a random.
- Generate an ECC key pair.
- Compute the Diffie-Hellman shared secret (pre-master secret)
- Retrieve the PSK-Identity.
- Compute the master secret with the PRF function

2.3.2 Applet Services to Run a (D)TLS1.3 handshake

For PKI schemes the applet provides:

- (Optional) Compute the early secret with the HKDF function
- Get a random.
- Put a public key in the applet store (issuer certificate deployment model).
- Verify a signature.
- Generate a signature.
- Generate an ECC key pair.
- Compute the Diffie-Hellman shared secret.
- (optional) Compute the handshake secret with the HKDF function.
- (optional) Compute the master secret with the HKDF function.

For the PSK-Plain scheme the Applet Provides:

- Get a random.
- Retrieve the PSK-Identity.
- Compute the early secret with the HKDF function.
- (optional) Compute the handshake secret with the HKDF function.
- (optional) Compute the master secret with the HKDF function.

For the PSK-ECDHE scheme the applet provides:

- Get a random.
- Retrieve the PSK-Identity.
- Compute the early secret with the HKDF function.
- Generate an ECC key pair.
- Compute the Diffie-Hellman shared secret.
- (optional) Compute the handshake secret with the HKDF function.
- (optional) Compute the master secret with the HKDF function.

2.3.3 Other Applet Services

For services that take more than one APDU to execute (signature generation; signature verification; put public key), the notion of ‘session’ is introduced. The applet may, as an implementer choice, support several sessions to allow the execution of several services in parallel (interleaved at APDU level).

Note: For implementations that can support multiple sessions, services that require a single APDU can be executed in the middle of a session.

2.4 An Example of How to Set the Applet Store

It is assumed that applet commissioning is from the IoT Security Server.

Note: Labels and identifiers are not illustrated.

2.4.1 Example for the PKI Scheme

For the ephemeral client **key pair**:

Key:

- Object access conditions (for keys): read to retrieve the public. None for the private key.
- Object state (for keys): public and private keys deactivated.
- Key type: NIST p256r1 or Brainpoolp256r1 (volatile).
- Key specific usage: general purpose.
- Algorithm for key agreement: ECKA.
- Cryptographic functions: key generation & Key agreement.

For the client credentials, with ECDSA or RSA-sign private key for the TLS handshake sequence signature & client's certificate:

Key (for Private Key):

- Object access conditions: none.
- Object state: activated (value downloaded).
- Key type: RSA2K or NIST p256r1 or Brainpoolp256r1 (persistent).
- Key specific usage: TLS-handshake Certificate Verify message.
- Cryptographic functions: signature.
- Algorithm for signature: RSASSA PKCS11 V1.5 or RSASSA PSS or ECDSA.
- Algorithms for hash: SHA-256 and optionally one or several from SHA-384, SHA-512.

File:

- Object access conditions (for files): read.
- Object state (for files): activated (value downloaded).
- File specific usage: X509v3 certificate.

For the issuer certificate model / server's ECDSA public key for server's ephemeral key verification extracted from the server certificate received during the handshake:

Key:

- Object access conditions (for keys): update.
- Object state (for keys): public key deactivated.
- Key type: NIST p256r1 or Brainpoolp256r1 (volatile).
- Key specific usage: general purpose & update (for the public part).
- Cryptographic functions: signature.
- Algorithm for signature: ECDSA.

For the issuer certificate model / issuer's ECDSA or RSA-sign public key for server's certificate verification. Note: The key can be extracted from the certificate by the IoT security server) & issuer's certificate.

Key:

- Object access conditions (for keys): none.
- Object state (for keys): public key activated with the key extracted from the issuer certificate.
- Key type: RSA2K or NIST p256r1 or Brainpoolp256r1 (persistent).
- Key specific usage: general purpose.
- Cryptographic functions: signature.
- Algorithm for signature: RSASSA PKCS11 V1.5 or RSASSA PSS or ECDSA.
- Algorithms for hash: SHA-256 and optionally one or several from SHA-384, SHA-512.

File:

- Object access conditions (for files): read.
- Object state (for files): activated (value downloaded).
- File specific usage: X509v3 certificate.

For the server certificate model / server's ECDSA public key for server's ephemeral key verification (Note: The key can be extracted from the certificate by the IoT security server & server's certificate).

Key:

- Object access conditions (for keys): none.
- Object state (for keys): public key activated (value downloaded).
- Key type: NIST p256r1 or Brainpoolp256r1 (persistent).
- Key specific usage: general purpose.
- Cryptographic functions: signature.
- Algorithm for signature: ECDSA.

File:

- Object access conditions (for files): read.
- Object state (for files): activated (value downloaded).
- File specific usage: X509v3 certificate.

2.4.2 Example for the PSK-Plain Scheme

Client credentials, with an HMAC capable key for master secret PRF generation (TLS v1.2).

Key:

- Object access conditions (for keys): none.
- Object state (for keys): activated, value set by the IoT Security Server.
- Key type: HMAC.
- Cryptographic functions: key derivation.
- Algorithm for key derivation: PRF.

File, optional, to store identification data for the server:

- Object access conditions (for files): read.
- Object state (for files): activated (value downloaded).
- File specific usage: general purpose.

Note: The applet store can provide identification data (e/UICC identification and key identification data) for key lookup on the server side. This identification data will be transferred to the server in the 'PSK identity' field of the client KeyExchange message.

2.5 General Data

2.5.1 Algorithms for Hash

b16 – b9	b8	b7	b6	b5	b4	b3	b2	b1	Description
-	-	-	-	-	-	-	-	1	SHA-256
-	-	-	-	-	-	-	1	-	SHA-384 – optional
-	-	-	-	-	-	1	-	-	SHA-512 – optional
x	x	x	x	x	x				RFU, those bits must be 0

Note: RSASSA PKCS#1 v1.5, RSASSA PSS, ECDSA, and HKDF can be used with different SHA algorithms

2.5.2 Algorithms for Signature

b8	b7	b6	b5	b4	b3	b2	b1	Description	Signature length (byte)
-	-	-	-	-	-	-	1	RSA with padding according to RSASSA PKCS#1 v1.5 - optional	Same as RSA key modulus length in bytes (256 bytes for 2048 bits keys)
-	-	-	-	-	-	1	-	RSA with padding according to RSASSA PSS - optional	Same as RSA key modulus length in bytes (256 bytes for a 2048 bit key)
-	-	-	-	-	1	-	-	ECDSA	Double as the ECC Key length in bytes (2*32=64 bytes for a 256 bit key)
x	x	X	x	x				RFU, those bits must be 0	

2.5.3 Algorithms for Key Agreement

b8	b7	b6	b5	b4	b3	b2	b1	Description	Pre-master secret length (byte)
-	-	-	-	-	-	-	1	ECKA (DL/ECKAS-DH1, IEEE 1363)	(See DH constants)
x	x	X	x	x	x	x		RFU, those bits must be 0	

2.5.4 Algorithms for Key Derivation

b8	b7	b6	b5	b4	b3	b2	b1	Description
-	-	-	-	-	-	-	1	PRF SHA-256 (rfc5246) - optional
-	-	-	-	-	-	1	-	HKDF (rfc5869)

x	x	X	x	x	x			RFU, those bits must be 0

2.5.5 Cryptographic Functions

b8	b7	b6	b5	b4	b3	b2	b1	Description
-	-	-	-	-	-	-	1	Signature (generation or verification)
0	0	0	0	0	0	1	-	Key generation
0	0	0	0	0	1	-	-	Key agreement
0	0	0	0	1	-	-	-	Key derivation - optional
x	x	x	x					RFU, those bits must be 0

2.5.6 DH Constants

Key Type	Pre-master secret length (Bytes)
NIST secp256r1	32
BrainpoolP256r1	32

2.5.7 ECC Public Key Format

Tag	Length	Value		
49h	Variable (1 byte)	**ECC Public Key Template**		
		Tag	**Length**	**Value**
		86h	2 x Z + 1	w = Point of the curve comprising the public key Format: 04 || X_w || Y_w (See ECC public key constants table)

2.5.8 ECC Public Key Constants

The value of Z and length of the data field varies according to the key length, as shown in the table below.

Key Length	Z (Bytes)	Maximum Length of Public Key template TLV (Bytes)
ELC 256	32	69 = (2 * Z) + 5

2.5.9 File Specific Usage

File specific usage	Description
1 byte	
01h	General purpose file
02h	X509v3 certificate storage
other	RFU

2.5.10 General Status Words

Status words listed below are applicable to all commands.

Status Word	Description
63CFh	The command execution is successful but a remote administration session has completed.
6400h	integrity Issue (applet store integrity issue detected)
6984h	Referenced data invalidated (remote administration session on-going, including update on targeted applet store object)
6989h	Remote administration session including update on targeted applet store object has completed, and the command execution has failed
6A84h	Insufficient memory
6D00h	Invalid instruction
6E00h	Invalid class
6F00h	Unknown error (other error)
9000h	Successful execution

2.5.11 Hash Algorithms Constant

Hash	Length of intermediate hash	*Block Length of Hash Algorithm*	Length of message digest result
SHA-256	32 bytes	*64 bytes*	32 bytes
SHA-384	64 bytes	*128 bytes*	48 bytes
SHA-512	64 bytes	*128 bytes*	64 bytes

2.5.12 Key Specific Usage

Key specific usage	Description
1 byte	
01h	General purpose key
02h	Key for Certificate Verify TLS1.2 handshake message
03h	Key for Certificate Verify TLS1.3 handshake message
other	RFU

2.5.13 Key Type

Key type value	Description
1 byte	
03h	RSA 2K - optional
13h	NIST secp256r1 (persistent)
14h	NIST secp256r1 (volatile)
23h	BrainpoolP256r1 (persistent)
24h	BrainpoolP256r1 (volatile)
A0h	HMAC capable key - optional
other	RFU

Note: Volatile keys are all deactivated upon applet deselection.

2.5.14 Mode of Operation for Signature Commands

Mode of operation for signature commands	Description
1 byte	
01h	Full text processing (the full text is hashed by the applet)
02h	Last block processing (the full text but the last block is hashed externally) *** This option is only available for signature generation ***
03h	Pad and sign processing (the full text is hashed externally)
other	RFU

2.5.15 Object Access Conditions

b8	b7	b6	b5	b4	b3	b2	b1	Description
-	-	-	-	-	-	-	1	Read
-	-	-	-	-	-	1	-	Update
x	x	x	x	x	x	-	-	RFU, those bits must be 0

2.5.16 Object State

b8	b7	b6		b5	b4	b3	b2	b1	Description
-	-	-		-	-	-	-	1	1 = Activated (filled) 0 = Deactivated (container empty or partially filled)
x	x	x		x	x	x	x		RFU, those bits must be 0

2.5.17 RSA Public Key Format

Tag	Length	Value		
48h	Variable (2 bytes)	**RSA Public Key Template**		
		Tag	**Length**	**Value**
		81h	0100h	Modulus (N)
		82h	01h–08h	Public Exponent (e)

2.6 Compute DH

2.6.1 Definition and Scope

The **Compute DH** command is used to generate a shared secret from a public key and a private key present in the applet store.

The command executes the Diffie-Hellman key agreement scheme with both keys and provides the caller with the shared secret.

Note: TLV must appear in the order they are listed.

2.6.2 Pre-Condition

Both public and private keys must be activated and must come from a different key pair. They must have the same key type and must be granted with a key agreement cryptographic function and with same ECKA key agreement algorithm.

2.6.3 Command Message

2.6.3.1 Command header

Code	Length (byte)	Description	Possible value
CLA	1	Class	100000xxb or 1100xxxxb: no secure messaging (b7 to b1 encoded as in ISO7816-4 2013)
INS	1	Compute DH command	46h
P1	1	Reference control parameter P1	00h
P2	1	Reference control parameter P2	00h
Lc	1	Length of the input data field	See data field in the command message
Le	1	Length of the response message	00h

2.6.3.2 Data Field in the command message

Tag	Length	Value	Presence
84h	01h-14h	Private key ID	Conditional (when the key is searched by ID)
85h	01h-14h	Public key ID	Conditional (when the key is searched by ID)
74h	01h-3Ch	Private key label	Conditional (when the key is searched by label)
75h	01h-3Ch	Public key label	Conditional (when the key is searched by label)

2.6.4 Response Message

2.6.4.1 Data Field returned in response message

Name	Length	Value	Presence
Shared secret	See DH constant table	-Value of the shared secret	Mandatory

2.6.4.2 Processing state returned in response message

This command may either return a general error condition as listed in "General Error Conditions" or one of the following error conditions.

Status Word	Description	Possible cause
6985h	Conditions of use not satisfied	• Key reference not found • The public key reference and private reference refer to same key pair • The public or private key are not activated • The public and private key don’t have same key type • The public or the private key don’t have key agreement

Status Word	Description	Possible cause
		cryptographic functions permission • The public or the private key don't have ECKA key agreement permission
6A80h	Incorrect data	• Issue in tags number or format
6A86h	Incorrect P1, P2	P1, P2 are not equal to 00h

2.7 Compute HKDF

2.7.1 Definition and scope

The **Compute HKDF** command is used to generate key material based on the HMAC-based key derivation function.

This command supports two modes of operation:

- HKDF-Extract general mode: the command is used to generate a pseudo random key. The input key material (IKM) comes as an input in command data.
- HKDF-Extract PSK-based: the command is used to generate a pseudo random key. The input key material (the secret) comes from a secret key of the applet store.

The hash function to use with the HKDF algorithm is indicated in input data.

Note: TLV must appear in the order they are listed.

2.7.2 Pre-Condition

The length of the salt must match with the length of the digest of the hash algorithm.

The secret key exists, is activated, it is granted with Key Derivation cryptographic function, and it is granted for HKDF key derivation algorithm (HKDF-Extract PSK-based).

2.7.3 Command Message

2.7.3.1 Command header

Code	Length (byte)	Description	Possible value
CLA	1	Class	100000xxb or 1100xxxxb: no secure messaging (b7 to b1 encoded as in ISO7816-4 2013)
INS	1	Compute HKDF command	4Ah
P1	1	Reference control parameter P1	HDKF generation mode • 00h: HKDF-extract general mode • 01h: HKDF-extract PSK-based
P2	1	Reference control parameter P2	00h
Lc	1	Length of the input data field	See data field in the command message
Le	1	Length of the response message	00h

2.7.3.2 Data Field in the command message

Tag	Length	Value	Presence
86h	01h-14h	Secret key ID	Conditional (when the key is searched by ID, And the HKDF-extract PSK-based generation mode is used)
76h	01h-3Ch	Secret key label	Conditional (when the key is searched by label, And the HKDF-extract PSK-based generation mode is used)
D1h	01h-40h	Secret	Conditional (when the HKDF-extract general mode is used)
D5h	20h-40h	Salt, length according to the digest of the hash function (See hash algorithms table)	Mandatory
91h	02h	Hash algorithm (See hash algorithms table)	Mandatory

2.7.4 Response Message

2.7.4.1 Data Field returned in response message

Name	Length	Value	Presence
Key material	20h – 40h	Pseudo-random value, length according to the digest of the hash function (See hash algorithms table)	Mandatory

2.7.4.2 Processing state returned in response message

This command may either return a general error condition as listed in "General Error Conditions" or one of the following error conditions.

Status Word	Description	Possible cause
6985h	Conditions of use not satisfied	• Secret key reference not found • The key is not activated • The secret key doesn’t have key derivation cryptographic functions permission • The secret key is not granted with HKDF key derivation algorithm
6A80h	Incorrect data	• Issue in tags number or format
6A86h	Incorrect P1, P2	• P1: unknown generation mode • P2: must be 00h

2.8 Compute PRF

2.8.1 Definition and scope

The **Compute PRF** command generates pseudo-random numbers based on the PRF function.

This command supports three modes of operation:

- General mode: the secret comes as an input data in command data.
- PSK-plain pre-master secret mode: the applet builds a pre-master secret in conformance with rfc4279, chapter 2, and uses it as the secret in the PRF computation. The PSK used for that pre-master secret computation comes from a secret key of the applet store.
- PSK-ECDHE pre-master secret mode: the applet builds a pre-master secret in conformance with rfc5489, chapter 2, and uses it as the secret in the PRF computation. The PSK used for that pre-master secret computation comes from a secret key of the applet store and the ECDH computation result comes as input in command data.

Note: TLV must appear in the order they are listed.

2.8.2 Pre-Condition

The key exists, is activated, it is granted with Key Derivation cryptographic function, and granted for PRF key derivation algorithm (PSK-plain and PSK-ECDHE pre-master secret mode).

2.8.3 Command Message

2.8.3.1 Command header

Code	Length (byte)	Description	Possible value
CLA	1	Class	100000xxb or 1100xxxxb: no secure messaging (b7 to b1 encoded as in ISO7816-4 2013)
INS	1	Compute PRF command	48h
P1	1	Reference control parameter P1	PRF generation mode • 00h: general mode • 01h: PSK-Plain pre-master secret mode • 02h: PSK-ECDHE pre-master secret mode
P2	1	Reference control parameter P2	00h
Lc	1	Length of the input data field	See data field in the command message
Le	1	Length of the response message	00h

2.8.3.2 Data Field in the command message

Tag	Length	Value	Presence
86h	01h-14h	Secret key ID	Conditional (when the key is searched by ID, And the PSK-Plain or PSK-ECDHE pre-master secret mode is used)
76h	01h-3Ch	Secret key label	Conditional (when the key is searched by label, And the PSK-Plain or PSK-ECDHE pre-master secret mode is used)
D1h	01h-40h	Secret	Conditional (when the general mode is used)

D4h	20h	ECDH computation result (See DH constant table)	Conditional (When the PSK-ECDHE pre-master secret mode is used)
D2h	01h-7Bh	Label and Seed *(byte string, concatenation of the label and seed parameters)*	Mandatory
D3h	01h	Pseudo-random length	Mandatory

2.8.4 Response Message

2.8.4.1 Data Field returned in response message

Name	Length	Value	Presence
PRF output	Variable	Pseudo-random value	Mandatory

2.8.4.2 Processing state returned in response message

This command may either return a general error condition as listed in "General Error Conditions" or one of the following error conditions.

Status Word	Description	Possible cause
6985h	Conditions of use not satisfied	• Secret key reference not found • The key is not activated • The secret key doesn't have key derivation cryptographic functions permission • The secret key is not granted with PRF SHA-256 key derivation algorithm
6A80h	Incorrect data	• Issue in tags number or format
6A86h	Incorrect P1, P2	• P1: unknown generation mode • P2: must be 00h

2.9 Compute signature – Init

2.9.1 Definition and scope

The **Compute Signature – Init** command opens a session to compute a signature. The command can also be used to cancel a signature computation session.

According to the table for mode of operation for signature commands (see section 2.5.14), three modes of operation are supported:

- Full text processing, meaning the full text is hashed by the applet before padding and signature computation.
- Last block processing, meaning the last block of the test is hashed by the applet before padding and signature computation.
- Pad and sign processing, meaning the hash on the full text is computed externally then transferred to the applet for padding and comparison with the value in the reference signature.

Sessions are all cancelled upon applet deselection.

Note: TLV must appear in the order they are listed.

2.9.2 Pre-Condition

The session number to open must be closed and the maximum number of sessions must not be reached (request for session opening).

The private key must exist, be activated and must be granted for signature with requested hash and signature algorithms (request for session opening).

The session to close must be opened and related to a Compute Signature operation (request for session cancellation).

2.9.3 Command Message

2.9.3.1 Command header

Code	Length (byte)	Description	Possible value
CLA	1	Class	100000xxb or 1100xxxxb: no secure messaging (b7 to b1 encoded as in ISO7816-4 2013)
INS	1	Compute Signature – Init command	2Ah
P1	1	Reference control parameter P1	Session request • 00h = open session • 01h = cancel session
P2	1	Reference control parameter P2	Session number
Lc	1	Length of the input data field	See data field in the command message

2.9.3.2 Data Field in the command message

Tag	Length	Value	Presence
84h	01h-14h	Private key ID	Conditional (when open session is requested, and when the key is searched by ID)
74h	01h-3Ch	Private key label	Conditional (when open session is requested, and the key is searched by label)
A1h	01h	Mode of operation for the signature command	Conditional (when open session is requested)
91h	02h	Hash algorithm (See hash algorithms table)	Conditional (when open session is requested)
92h	01h	Signature algorithm (See signature algorithms table)	Conditional (when open session is requested)

2.9.4 Response Message

The response message returns SW1 and SW2 status word.

2.9.4.1 Processing state returned in response message

This command may either return a general error condition as listed in "General Error Conditions" or one of the following error conditions.

Status Word	Description	Possible cause
6700h	Wrong length (incorrect Lc)	Lc is not 00h (when close session is requested)
6985h	Conditions of use not satisfied	• Key reference not found • Key not activated • The private key doesn't have signature function permission • Unsupported signature or hash algorithm • Unsupported signature mode of operation
6989h	Maximum capacity reached	Maximum number of sessions reached
6A80h	Incorrect data	Issue in tags number or format
6A86h	Incorrect P1, P2	• P1: out of bound session request • P2: session number incompatible with the session request (ex., request to open a session already opened, request to close a session for some other service)

2.10 Compute signature – Update

2.10.1 Definition and scope

The **Compute Signature – Update** command is used to provide the applet with reference data to compute and return a signature to the caller**.**

According to the mode of operation for signature commands, reference data are as below:

- The full text or –exclusive- the final hash or –exclusive- the last block to hash, the intermediate hash, and the number of bytes already hashed.

With the 'full text' mode, several APDU might be necessary to provide reference data. In that case, it is expected that all but the last data block are filled with 255 bytes of data.

First APDU with more incoming data are indicated with P1.b8 = 0 & P1.b0 = 0. The last APDU with last incoming data is indicated with P1.b8 = 1 & P1.b0. This APDU returns output data and a 6300h SW whenever further output data can be fetched. Subsequent outgoing data is indicated with P1.b8 = 1 & P1.b0. All along that sequence CLA, INS and P2 bytes must be keep the same value.

The signature is returned and is formatted as below:

- RSA: the digital signature is unformatted.
- ECDSA: the digital signature is in the format r||s where r and s are integers and coordinates the points on the elliptic curve that correspond to the signature value.

Note: TLV must appear in the order they are listed.

2.10.2 Pre-Condition

The session is opened.

2.10.3 Command Message

2.10.3.1 Command header

<table>
<tr><th>Code</th><th>Length (byte)</th><th>Description</th><th>Possible value</th></tr>
<tr><td>CLA</td><td>1</td><td>Class</td><td>100000xxb or 1100xxxxb: no secure messaging (b7 to b1 encoded as in ISO7816-4 2013).</td></tr>
<tr><td>INS</td><td>1</td><td>Compute Signature – Update command</td><td>2Bh</td></tr>
<tr><td>P1</td><td>1</td><td>Reference control parameter P1</td><td>
<table>
<tr><th>b8</th><th>b7</th><th>b6</th><th>b5</th><th>b4</th><th>b3</th><th>b2</th><th>b1</th><th>Meaning</th></tr>
<tr><td>0</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td><td>More incoming data</td></tr>
<tr><td>1</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td><td>Last incoming data</td></tr>
<tr><td>x</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td><td>0</td><td>First outgoing data</td></tr>
<tr><td>x</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td><td>1</td><td>More outgoing data</td></tr>
<tr><td>-</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>-</td><td>RFU (must be 0)</td></tr>
</table>
</td></tr>
<tr><td>P2</td><td>1</td><td>Reference control parameter P2</td><td>Session number</td></tr>
<tr><td>Lc</td><td>1</td><td>Length of the input data field</td><td>See data field in the command message</td></tr>
<tr><td>Le</td><td>1</td><td>Length of the response message</td><td>00h</td></tr>
</table>

2.10.3.2 Data Field in the command message

Tag	Length	Value	Presence
9Ah	01h – 80h	Last block to hash (See the hash algorithms constant table)	Conditional (when the last block is hashed by the applet)
9Bh	000000h - FFFFFFh	Data for which signature generation is requested	Conditional (when the full text is hashed to the applet)
9Eh	20h – 40h	Final hash (See the hash algorithms constant table)	Conditional (when the final hash is provided to the applet)
9Ch	20h – 40h	Intermediate hash (See the hash algorithm constant table)	Conditional (when the last block is hashed in the applet)
9Dh	04h	Number of bytes already hashed	Conditional (when the last block is hashed in the applet)

2.10.4 Response Message

2.10.4.1 Data Field returned in response message

Name	Length	Value			Presence
		Tag	Length	Value	
Computed signature	variable	33h	From 40h for ECDSA 256 bits up to 0100h for RSA 2048 bits	Signature data (See the algorithms for signature table)	Mandatory

2.10.4.2 Processing state returned in response message

This command may either return a general error condition as listed in "General Error Conditions" or one of the following error conditions.

Status Word	Description	Possible cause
6300h	More data available	More data are available and can be retrieved with the more outgoing data mode
6700h	Wrong length (incorrect Lc)	Lc for a non-final block is not equal to 255
6985h	Conditions of use not satisfied	Length of the data to hash, length of final hash, length of the intermediate hash, length of the last block to hash, or number of bytes already hashed are not compatible with the requested operation
6A80h	Incorrect data	Issue in tags number or format
6A86h	Incorrect P1, P2	• P1: unknown data incoming or data outgoing mode / CLA, INS, P1 bytes don't match with incoming or outgoing modes • P2: the session is not opened

2.11 Generate Key Pair

2.11.1 Definition and scope

The **Generate Key Pair** command is used to generate an asymmetric key pair. Upon successive execution both public and private keys are updated (and activated) in the store with their respective value and the public key is returned to the caller.

Note: The private key reference is used to identify both parts of the key pair.

Note: The type/length of the key pair to generate is known thanks to the key type associated to it.

Note: As the applet targets ECDHE based cipher suites the applet needs to support ECC key pair generation (not RSA).

Note: This operation may modify the key ID.

Note: TLV must appear in the order they are listed.

2.11.2 Pre-Condition

The target key pair (recognized because the public key and private key have same label) must exist in the key store and must not be in use in a cryptographic session. Both public

and private keys must be granted with key generation cryptographic function and the public key with read object access condition.

2.11.3 Command Message

2.11.3.1 Command header

Code	Length (byte)	Description	Possible value
CLA	1	Class	100000xxb or 1100xxxxb: no secure messaging (b7 to b1 encoded as in ISO7816-4 2013)
INS	1	Generate Key Pair	B9h
P1	1	Reference control parameter P1	00h
P2	1	Reference control parameter P2	00h
Lc	1	Length of the input data field	See data field in the command message
Le	1	Length of the response message	00h

2.11.3.2 Data Field in the command message

Tag	Length	Value	Presence
84h	01h-14h	Private key Id	Conditional (when the key is searched by ID)
74h	01h-3Ch	Private Key label	Conditional (when the key is searched by label)

2.11.4 Response Message

2.11.4.1 Data Field returned in response message

Name	Length	Value			Presence
Private key identifier	03h-16h bytes	Tag	Length	Value	Mandatory
		84h	01h-14h	Private key Id	
Public key identifier	03h-16h bytes	Tag	Length	Value	Mandatory
		85h	01h-14h	Public key Id	
Public key data	Variable	Tag	Length	Value	Mandatory
		34h	45h	ECC public key (See key format tables)	

2.11.4.2 Processing state returned in response message

This command may either return a general error condition as listed in "General Error Conditions" or one of the following error conditions.

Status Word	Description	Possible cause
6985h	Conditions of use not satisfied	• The private key could not be found • The public key associated to the private could not be found • Keys don't have key generation cryptographic function permission • The public key doesn't have read object access condition • Keys are is in use by another session
6A80h	Incorrect data	Issue in tags number or format
6A86h	Incorrect P1, P2	P1, P2 are not equal to 00h
6F00h	Unknown error	Key generation has failed generating the key pair

2.12 Get data – application

2.12.1 Definition and scope

The Get data - Application command returns information about the applet and its capacity.

Note: TLV must appear in the order they are listed.

2.12.2 Pre-Condition

Void

2.12.3 Command Message

2.12.3.1 Command header

Code	Length (byte)	Description	Possible value
CLA	1	Class	100000xxb or 1100xxxxb: no secure messaging (b7 to b1 encoded as in ISO7816-4 2013)
INS	1	Get Data command	CBh
P1	1	Reference control parameter P1	00h = get data – application
P2	1	Reference control parameter P2	00h
Le	1	Length of the response message	44h - Expected application information

2.12.4 Response Message

2.12.4.1 Data Field returned in response message

Name	Length	Value	Presence
SIM Alliance version	03h bytes	Tag: 10h; Length: 01h; Value: Version of the SIM Alliance applet specification	Mandatory
Applet proprietary identifier	22h bytes	Tag: 11h; Length: 20h; Value: 32 byte proprietary applet identifier	Mandatory
Max number of files	03h bytes	Tag: B1h; Length: 01h; Value: 01h – FFh = maximum number of files	Mandatory

Max number of private keys	03h bytes		**Tag**	**Length**	**Value**		Mandatory
			B2h	01h	01h – FFh = maximum number of private keys		
Max number public keys	03h bytes		**Tag**	**Length**	**Value**		Mandatory
			B3h	01h	01h – FFh = maximum number of public keys		
Max number secret keys	03h bytes		**Tag**	**Length**	**Value**		Mandatory
			B4h	01h	01h – FFh = maximum number of secret keys		
Cryptographic functions	03h bytes		**Tag**	**Length**	**Value**		Mandatory
			90h	01h	Supported cryptographic functions coded as a combination of bits (See cryptographic function table)		
Supported algorithms for hash	04h bytes		**Tag**	**Length**	**Value**		Mandatory
			91h	02h	Supported algorithms for hash coded as a combination of bits (See algorithms for hash table)		
Supported algorithms for signature	03h bytes		**Tag**	**Length**	**Value**		Mandatory
			92h	01h	Supported algorithms for signature coded as a combination of bits (See algorithms for signature table)		
Supported algorithms for key agreement	03h bytes		**Tag**	**Length**	**Value**		Mandatory
			93h	01h	Supported algorithms for key agreement coded as a combination of bits (See algorithms for key agreement table)		
Supported algorithms for key derivation	03h bytes		**Tag**	**Length**	**Value**		Mandatory
			94h	01h	Supported algorithms for key derivation coded as a combination of bits (See algorithms for key derivation table)		
Maximum number of sessions	03h bytes		**Tag**	**Length**	**Value**		Mandatory
			B7h	01h	00h - FFh = max number of concurrent sessions		

2.12.4.2 Processing state returned in response message

This command may either return a general error condition as listed in "General Error Conditions" or one of the following error conditions.

Status Word	Description	Possible cause
6700h	Wrong length (incorrect Le)	See Le possible value
6A86h	Incorrect P1, P2	• P1: unknown get data option • P2: different from 00h

2.13 Get data – file

2.13.1 Definition and scope

The **Get Data - File** command retrieves all information associated to a file in the applet store.

Note: TLV must appear in the order they are listed.

2.13.2 Pre-Condition

The targeted file must exist.

2.13.3 Command Message

2.13.3.1 Command header

Code	Length (byte)	Description	Possible value
CLA	1	Class	100000xxb or 1100xxxxb: no secure messaging (b7 to b1 encoded as in ISO7816-4 2013).
INS	1	Get Data command	CBh
P1	1	Reference control parameter P1	C3h = get data - file
P2	1	Reference control parameter P2	00h
Lc	1	Length of the input data field	See data field in the command message
Le	1	Length of the response message	00h

2.13.3.2 Data Field in the command message

Tag	Length	Value	Presence
73h	01h-3Ch	File label	Conditional (when the file is searched by label)
83h	01h-14h	File ID	Conditional (when the file is searched by ID)

2.13.4 Response Message

2.13.4.1 Data Field returned in response message

Name	Length	Value			Presence
		Tag	Length	Value	
File information structure	variable	C3h	variable	See file information structure in Get Data – Object List	Mandatory

2.13.4.2 Processing state returned in response message

This command may either return a general error condition as listed in "General Error Conditions" or one of the following error conditions.

Status Word	Description	Possible cause
6A80h	Incorrect data	Issue in tags number or format
6A82h	File not found	File reference not found

6A86h	Incorrect P1, P2	• P1: unknown get data option • P2: must be 00h

2.14 Get data – object list

2.14.1 Definition and scope

The **Get Data - Object List** command list all objects (files, key pairs, secret keys) and their attributes present in the applet store.

Objects are listed one after the other. They are grouped together in the same response message. A response message only contains the complete object information structures.

The ‘data outgoing mode’ (see P2) indicates where the first objects shall be fetched (first outgoing data) or where the next objects shall be fetched (after first outgoing data or more outgoing data). The presence of objects to retrieve is indicated with the SW 6300h. From that SW fetching subsequent objects requires the next command to be provided with the exact same CLA, INS and P1 bytes but P2.

Note: The order objects are returned is implementation dependent but TLV must appear in the order they are listed.

2.14.2 Pre-Condition

Void

2.14.3 Command Message

2.14.3.1 Command header

Code	Length (byte)	Description	Possible value
CLA	1	Class	100000xxb or 1100xxxxb: no secure messaging (b7 to b1 encoded as in ISO7816-4 2013).
INS	1	Get Data command	CBh
P1	1	Reference control parameter P1	01h = get data – object list
P2	1	Reference control parameter P2	Data outgoing mode • 00h = first outgoing data • 01h = more outgoing data
Le	1	Length of the response message	00h

2.14.4 Response Message

2.14.4.1 Data Field returned in response message

Name	Length	Value	Presence
Private key list	N * private key information structure length	Tag: C1h; Length: variable; Value: Private key information structure	Conditional (as many as private key in the applet store)
Pubic key list	N * public key information structure length	Tag: C2h; Length: variable; Value: Public key information structure	Conditional (as many as public key in the applet store)
File list	N * file attributes structure length	Tag: C3h; Length: variable; Value: File information structure	Conditional (as many as files in the applet store)
Secret key list	N * secret key information structure length	Tag: C4h; Length: variable; Value: Secret key information structure	Conditional (as many as secret key in the applet store)

2.14.4.2 Private key information structure

Name	Length	Value	Presence
Private key label	03h – 3Eh bytes	Tag: 74h; Length: 01h – 3Ch; Value: Private key Label	Conditional (if a label is associated to the private key)
Private key identifier	03h -16h bytes	Tag: 84h; Length: 01h-14h; Value: Private key Id	Mandatory
Object access conditions (for the private key)	03h bytes	Tag: 60h; Length: 01h; Value: Objects access conditions coded as a combination of bits (See object access conditions table)	Mandatory
Object state (for the private key)	03h bytes	Tag: 4Ah; Length: 01h; Value: (See object state table)	Mandatory
Key type	03h bytes	Tag: 4Bh; Length: 01h; Value: (See key type table)	Mandatory
Key specific usage	03h bytes	Tag: 4Eh; Length: 01h; Value: (See key specific usage table)	Mandatory
Cryptographic functions	03h bytes	Tag: 61h; Length: 01h; Value: Cryptographic functions coded as a combination of bits (See cryptographic functions table)	Mandatory

Supported algorithms for signature (generation)	03h bytes		**Tag**	**Length**	**Value**		Conditional (When the key is used for digital signature)
			92h	01h	Supported algorithms coded as a combination of bits (See algorithms for signature table)		
Supported algorithms for hash	04h bytes		**Tag**	**Length**	**Value**		Conditional (When the key is used for digital signature)
			91h	02h	Supported algorithms coded as a combination of bits (See hash algorithms table)		
Supported algorithms for key agreement	03h bytes		**Tag**	**Length**	**Value**		Conditional (When the key is used for key agreement)
			6Fh	01h	Supported algorithms coded as a combination of bits (See algorithms for key agreement table)		

2.14.4.3 Public key information structure

Name	Length	Value					Presence
Public key label	03h – 3Eh bytes		**Tag**	**Length**	**Value**		Conditional (if a label is associated to the public key)
			75h	01h – 3Ch	Public key Label		
Public key identifier	03h -16h bytes		**Tag**	**Length**	**Value**		Mandatory
			85h	01h-14h	Public key Id		
Object access conditions (for the public key)	03h bytes		**Tag**	**Length**	**Value**		Mandatory
			60h	01h	Objects access conditions coded as a combination of bits (See object access conditions table)		
Object state (for the public key)	03h bytes		**Tag**	**Length**	**Value**		Mandatory
			4Ah	01h	(See object state table)		
Key type	03h bytes		**Tag**	**Length**	**Value**		Mandatory
			4Bh	01h	(See key type table)		
Key specific usage	03h bytes		**Tag**	**Length**	**Value**		Mandatory
			4Eh	01h	(See key specific usage table)		
Cryptographic function	03h bytes		**Tag**	**Length**	**Value**		Mandatory
			61h	01h	Cryptographic function coded as a combination of bits (See cryptographic functions table)		
Supported algorithms for signature (verification)	03h bytes		**Tag**	**Length**	**Value**		Conditional (When the key is used for digital signature)
			92h	01h	Supported algorithms coded as a combination of bits (See algorithms for signature table)		
Supported algorithms for hash	04h bytes		**Tag**	**Length**	**Value**		Conditional (When the key is used for digital signature)
			91h	02h	Supported algorithms coded as a combination of bits (See hash algorithms table)		

Supported algorithms for key agreement	03h bytes		**Tag**	**Length**	**Value**		Conditional (When the key is used for key agreement)
			6Fh	01h	Supported algorithms coded as a combination of bits (See algorithms for key agreement table)		

2.14.4.4 File information structure

Name	**Length**	**Value**			**Presence**
File label	03h – 3Eh bytes	**Tag**	**Length**	**Value**	Conditional (if a label is associated to the file)
		73h	01h-3Ch	File label	
File ID	03h -16h bytes	**Tag**	**Length**	**Value**	Mandatory
		83h	01h-14h	File ID	
Object (file) access conditions	03h bytes	**Tag**	**Length**	**Value**	Mandatory
		60h	01h	Objects access conditions coded as a combination of bits (See objects access conditions table)	
Object state (for the file)	03h bytes	**Tag**	**Length**	**Value**	Mandatory
		4Ah	01h	Object state (See object state table)	
File specific usage	03h bytes	**Tag**	**Length**	**Value**	Mandatory
		21h	01h	File specific usage (See file specific usage table)	
File size	04h bytes	**Tag**	**Length**	**Value**	Mandatory
		20h	02h	File size	

2.14.4.5 Secret key information structure

Name	**Length**	**Value**			**Presence**
Secret key label	03h – 3Eh bytes	**Tag**	**Length**	**Value**	Conditional (if a label is associated to the secret key)
		76h	01h – 3Ch	Secret key Label	
Secret key identifier	03h -16h bytes	**Tag**	**Length**	**Value**	Mandatory
		86h	01h-14h	Secret key Id	
Object access conditions (for the secret key)	03h bytes	**Tag**	**Length**	**Value**	Mandatory
		60h	01h	Objects access conditions coded as a combination of bits (See object access conditions table)	
Object state (for the secret key)	03h bytes	**Tag**	**Length**	**Value**	Mandatory
		4Ah	01h	(See object state table)	
Key type	03h bytes	**Tag**	**Length**	**Value**	Mandatory
		4Bh	01h	(See key type table)	

Cryptographic functions	03h bytes		**Tag**	**Length**	**Value**		Mandatory
			61h	01h	Cryptographic functions coded as a combination of bits (See cryptographic functions table)		
Supported algorithms for key derivation	03h bytes		**Tag**	**Length**	**Value**		Conditional (When the key is used for key derivation)
			94h	01h	Supported algorithms coded as a combination of bits (See algorithms for signature table)		

2.14.4.6 Processing state returned in response message

This command may either return a general error condition as listed in "General Error Conditions" or one of the following error conditions.

Status Word	Description	Possible cause
6300h	More data available	More data are available and can be retrieved with the more outgoing data mode
6A86h	Incorrect P1, P2	• P1: unknown get data option • P2: unknown data outgoing mode / CLA, INS, P1 bytes don't match with a more data outgoing mode

2.15 Get data – private key information

2.15.1 Definition and scope

The **Get data – private key** command retrieves all information associated to a private key.

Note: TLV must appear in the order they are listed.

2.15.2 Pre-Condition

The targeted key must exist.

2.15.3 Command Message

2.15.3.1 Command header

Code	Length (byte)	Description	Possible value
CLA	1	Class	100000xxb or 1100xxxxb: no secure messaging (b7 to b1 encoded as in ISO7816-4 2013).
INS	1	Get Data command	CBh
P1	1	Reference control parameter P1	C1h = get data – private key
P2	1	Reference control parameter P2	00h
Lc	1	Length of the input data field	See data field in the command message
Le	1	Length of the response message	00h

2.15.3.2 Data Field in the command message

Tag	Length	Value	Presence
74h	01h-3Ch	Private key label	Conditional (when the key is searched by label)
84h	01h-14h	Private key ID	Conditional (when the key is searched by ID)

2.15.4 Response Message

2.15.4.1 Data Field returned in response message

Name	Length	Value			Presence
		Tag	Length	Value	
Private key information structure	variable	C1h	variable	See private key information structure in Get Data – Object List	Mandatory

2.15.4.2 Processing state returned in response message

This command may either return a general error condition as listed in "General Error Conditions" or one of the following error conditions.

Status Word	Description	Possible cause
6985h	Conditions of use not satisfied	Key reference not found
6A80h	Incorrect data	Issue in tags number or format
6A86h	Incorrect P1, P2	• P1: unknown get data option • P2: must be 00h

2.16 Get data – public key

2.16.1 Definition and scope

The **Get Data – Public Key** command retrieves all information associated to a public key.

Note: TLV must appear in the order they are listed.

2.16.2 Pre-Condition

The targeted key must exist.

2.16.3 Command Message

2.16.3.1 Command header

Code	Length (byte)	Description	Possible value
CLA	1	Class	100000xxb or 1100xxxxb: no secure messaging (b7 to b1 encoded as in ISO7816-4 2013).
INS	1	Get Data command	CBh
P1	1	Reference control parameter P1	C2h = get data – public key

P2	1	Reference control parameter P2	00h
Lc	1	Length of the input data field	See data field in the command message
Le	1	Length of the response message	00h

2.16.3.2 Data Field in the command message

Tag	Length	Value	Presence
75h	01h-3Ch	Public key label	Conditional (when the key is searched by label)
85h	01h-14h	Public key ID	Conditional (when the key pair is searched by ID)

2.16.4 Response Message

2.16.4.1 Data Field returned in response message

Name	Length	Value			Presence
		Tag	Length	Value	
Public key information structure	variable	C2h	variable	See public key information structure in Get Data – Object List	Mandatory

2.16.4.2 Processing state returned in response message

This command may either return a general error condition as listed in "General Error Conditions" or one of the following error conditions.

Status Word	Description	Possible cause
6985h	Conditions of use not satisfied	Key reference not found
6A80h	Incorrect data	Issue in tags number or format
6A86h	Incorrect P1, P2	• P1: unknown get data option • P2: must be 00h

2.17 Get data – secret key information

2.17.1 Definition and scope

The **Get Data – Secret Key** command retrieves all information associated to a secret key.

Note: TLV must appear in the order they are listed.

2.17.2 Pre-Condition

The targeted key must exist.

2.17.3 Command Message

2.17.3.1 Command header

Code	Length (byte)	Description	Possible value
CLA	1	Class	100000xxb or 1100xxxxb: no secure messaging (b7 to b1 encoded as in ISO7816-4 2013).
INS	1	Get Data command	CBh
P1	1	Reference control parameter P1	C4h = get data – secret key
P2	1	Reference control parameter P2	00h
Lc	1	Length of the input data field	See data field in the command message
Le	1	Length of the response message	00h

2.17.3.2 Data Field in the command message

Tag	Length	Value	Presence
76h	01h-3Ch	Secret key label	Conditional (when the key is searched by label)
86h	01h-14h	Secret key ID	Conditional (when the key pair is searched by ID)

2.17.4 Response Message

2.17.4.1 Data Field returned in response message

Name	Length	Value			Presence
		Tag	Length	Value	
Secret key information structure	variable	C4h	variable	See secret key information structure in Get Data – Object List	Mandatory

2.17.4.2 Processing state returned in response message

This command may either return a general error condition as listed in "General Error Conditions" or one of the following error conditions.

Status Word	Description	Possible cause
6985h	Conditions of use not satisfied	Key reference not found
6A80h	Incorrect data	Issue in tags number or format
6A86h	Incorrect P1, P2	• P1: unknown get data option • P2: must be 00h

2.18 Get random

2.18.1 Definition and scope

Get Random command returns a random value. Its length can extend from 1 byte up to 256 bytes.

2.18.2 Pre-Condition

Void.

2.18.3 Command Message

2.18.3.1 Command header

Code	Length (byte)	Description	Possible value
CLA	1	Class	100000xxb or 1100xxxxb: no secure messaging (b7 to b1 encoded as in ISO7816-4 2013).
INS	1	Get random command	84h
P1	1	Reference control parameter P1	00h
P2	1	Reference control parameter P2	00h
Le	1	Length of the response message	Variable, see the response message (00h means 256 bytes expected)

2.18.4 Response Message

2.18.4.1 Data Field returned in response message

Name	Length	Value	Presence
Card random	variable	Random value	Mandatory

2.18.4.2 Processing state returned in response message

This command may either return a general error condition as listed in "General Error Conditions" or one of the following error conditions.

Status Word	Description	Possible cause
6A86h	Incorrect P1, P2	P1, P2 are not equal to 00h

2.19 Put public key – Init

2.19.1 Definition and scope

The **Put Public Key – Init** command opens a session to update a public key. The command can also be used to cancel a put public key session.

The command deactivates (object state) the public key and the private key that may be associated with it. This may impact both key identifiers.

Sessions are all cancelled upon applet deselection.

Note: TLV must appear in the order they are listed.

2.19.2 Pre-Condition

The session number to open must be closed and the maximum number of sessions must not be reached (request for session opening).

The target public key must exist, must be granted for update (object access conditions) and must not be in use in a cryptographic session. Should a private key be associated to the public key (same label), the private key must be granted for updates (object access conditions) and must not be in use in a cryptographic session"

The session to close must be opened and related to a Put public key operation (request for session cancellation).

2.19.3 Command Message

2.19.3.1 Command header

Code	Length (byte)	Description	Possible value
CLA	1	Class	100000xxb or 1100xxxxb: no secure messaging (b7 to b1 encoded as in ISO7816-4 2013).
INS	1	Put Public Key – Init command	24h
P1	1	Reference control parameter P1	Session request ▪ 00h = open session ▪ 01h = cancel session
P2	1	Reference control parameter P2	Session number
Lc	1	Length of the input data field	See data field in the command message

2.19.3.2 Data Field in the command message

Tag	Length	Value	Presence
85h	01h-14h	Public key ID	Conditional (when open session is requested, and when the key is searched by ID)
75h	01h – 3Ch	Public key label	Conditional (when open session is requested, and the key is searched by label)

2.19.4 Response Message

The response message returns the SW1 and SW2 status word.

2.19.4.1 Processing state returned in response message

This command may either return a general error condition as listed in "General Error Conditions" or one of the following error conditions.

Status Word	Description	Possible cause
6700h	Wrong length (incorrect Lc)	Lc is not 00h (when close session is requested)
6985h	Conditions of use not satisfied	▪ Key reference not found ▪ The public key doesn't have update object access conditions ▪ The key is in use by another session
6989h	Maximum capacity reached	Maximum number of sessions reached
6A80h	Incorrect data	Issue in tags number or format

6A86h	Incorrect P1, P2	• P1: out of bound session request • P2: session number incompatible with the session request (ex., request to open a session already opened, request to close a session for some other service)

2.20 Put public key – update

2.20.1 Definition and scope

The **Put Public Key – Update** command is used to load a public (ECC or RSA) key in the applet store.

The command is used as many times as necessary, to provide the applet with all key chunks.

When more than a command is necessary to provide the key, it is expected that all but the last data block are filled with 255 bytes of data.

Upon successful loading the public key is activated.

Note: TLV must appear in the order they are listed.

2.20.2 Pre-Condition

The session is opened.

2.20.3 Command Message

2.20.3.1 Command header

<table>
<tr><th>Code</th><th>Length (byte)</th><th>Description</th><th>Possible value</th></tr>
<tr><td>CLA</td><td>1</td><td>Class</td><td>100000xxb or 1100xxxxb: no secure messaging (b7 to b1 encoded as in ISO7816-4 2013).</td></tr>
<tr><td>INS</td><td>1</td><td>Put Public Key – Update command</td><td>D8h</td></tr>
<tr><td>P1</td><td>1</td><td>Reference control parameter P1</td><td>
<table>
<tr><th>b8</th><th>b7</th><th>b6</th><th>b5</th><th>b4</th><th>b3</th><th>b2</th><th>b1</th><th>Meaning</th></tr>
<tr><td>0</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td><td>More incoming data</td></tr>
<tr><td>1</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td><td>Last incoming data</td></tr>
<tr><td>-</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>RFU (must be 0)</td></tr>
</table>
</td></tr>
<tr><td>P2</td><td>1</td><td>Reference control parameter P2</td><td>Session number</td></tr>
<tr><td>Lc</td><td>1</td><td>Length of the input data field</td><td>See data field in the command message</td></tr>
</table>

2.20.3.2 Data Field in the command message

Tag	Length	Value	Presence
34h	variable	RSA or ECC public key (See key format tables)	Mandatory

2.20.4 Response Message

2.20.4.1 Processing state returned in response message

This command may either return a general error condition as listed in "General Error Conditions" or one of the following error conditions.

Status Word	Description	Possible cause
6700h	Wrong length (incorrect Lc)	Lc for a non-final command is not equal to 255
6985h	Conditions of use not satisfied	The public key does not match with expected key type
6A80h	Incorrect data	Issue in tags number or format
6A86h	Incorrect P1, P2	▪ P1: the value differs from last incoming data or more incoming data ▪ P2: the session is not opened

2.21 Read file

2.21.1 Definition and scope

The **Read File** command reads the contents of an EF from the start offset and returns 256 bytes or less if it reaches the end file

Note: EF are all located under the MF (there is no subdirectory).

Note: TLV must appear in the order they are listed.

2.21.2 Pre-Condition

The targeted file must exist, must be activated, and must be granted with read object access condition.

2.21.3 Command Message

2.21.3.1 Command header

Code	Length (byte)	Description	Possible value
CLA	1	Class	100000xxb or 1100xxxxb: no secure messaging (b7 to b1 encoded as in ISO7816-4 2013).
INS	1	Read File command	B0h
P1	1	Reference control parameter P1	Offset in the file
P2	1	Reference control parameter P2	
Lc	1	Length of the input data field	See data field in the command message
Le	1	Length of the response message	00h

2.21.3.2 Data Field in the command message

Tag	Length	Value	Presence
83h	01h-14h	File ID	Conditional (when the file is selected by identifier)
73h	01h – 3Ch	File label	Conditional (when the file is selected by label)

2.21.4 Response Message

2.21.4.1 Data Field returned in response message

Name	Length	Value	Presence
File content	variable	Data read	Mandatory

2.21.4.2 Processing state returned in response message

This command may either return a general error condition as listed in "General Error Conditions" or one of the following error conditions.

Status Word	Description	Possible cause
6981h	Command incompatible with file structure	Given offset is outside EF limits
6985h	Conditions of use not satisfied	• Read is not granted • The file is deactivated
6A80h	Incorrect data	Issue in tags number or format
6A82h	File not found	File reference not found
6A86h	Incorrect P1, P2	Offset is outside EF limits

2.22 Read public key

2.22.1 Definition and scope

The **Read public key** command retrieves a public key from the applet store.

The ‘data outgoing mode’ (see P2) indicates whether the first 255 byes shall be fetched (first outgoing data) or whether the next 255 bytes shall be fetched (after first outgoing data or more outgoing data). The presence of data to retrieve is indicated with the SW 6300h. From that SW fetching subsequent data requires next command to be provided with exact same CLA, INS and P1 bytes but P2.

Note: TLV must appear in the order they are listed.

2.22.2 Pre-Condition

The target key must exist, must be activated, and must be granted with read object access condition.

2.22.3 Command Message

2.22.3.1 Command header

Code	Length (byte)	Description	Possible value
CLA	1	Class	100000xxb or 1100xxxxb: no secure messaging (b7 to b1 encoded as in ISO7816-4 2013).
INS	1	Read Key command	CDh
P1	1	Reference control parameter P1	00h
P2	1	Reference control parameter P2	Data outgoing mode • 00h = first outgoing data • 01h = more outgoing data
Lc	1	Length of the input data field	See data field in the command message
Le	1	Length of the response message	00h

2.22.3.2 Data Field in the command message

Tag	Length	Value	Presence
85h	01h-14h	Public key ID	Conditional (when the public key is selected by identifier)
75h	01h – 3Ch	Public key label	Conditional (when the public key is selected by label)

2.22.4 Response Message

2.22.4.1 Data Field returned in response message

Name	Length	Value			Presence
		Tag	Length	Value	
Public key data	variable	34h	Variable	RSA or ECC public key (See key format tables)	Mandatory

2.22.4.2 Processing state returned in response message

This command may either return a general error condition as listed in "General Error Conditions" or one of the following error conditions.

Status Word	Description	Possible cause
6300h	More data available	More data are available and can be retrieved with the more outgoing data mode
6985h	Conditions of use not satisfied	• The public key could not be found • The public key doesn't have read object access condition • The public key is deactivated
6A80h	Incorrect data	Issue in tags number or format

6A86h	Incorrect P1, P2	• P1: if not equal to 00h • P2: unknown data outgoing mode / CLA, INS, P1 bytes don't match with a more data outgoing mode

2.23 Verify signature – init

2.23.1 Definition and scope

The **Verify Signature – Init** command opens a session to verify a signature. The command can also be used to cancel a signature verification session.

According to the table for mode of operation for signature commands (see section 2.5.14), two modes of operation are supported:

- Full text processing, meaning the full text is hashed by the applet before padding and comparison with the value in the reference signature.
- Pad and sign processing, meaning the hash of the full text is computed externally then transferred to the applet for padding and comparison with the value in the reference signature.

Sessions are all cancelled upon applet deselection.

Note: TLV must appear in the order they are listed.

2.23.2 Pre-Condition

The session number to open must be closed and the maximum number of sessions must not be reached (request for session opening).

The public key must exist, be activated, and must be granted for signature with requested hash and signature algorithm (request for session opening).

The session to close must be opened and related to a Verify Signature operation (request for session cancellation).

2.23.3 Command Message

2.23.3.1 Command header

Code	Length (byte)	Description	Possible value
CLA	1	Class	100000xxb or 1100xxxxb: no secure messaging (b7 to b1 encoded as in ISO7816-4 2013).
INS	1	Verify Signature – Init command	2Ch
P1	1	Reference control parameter P1	Session request ▪ 00h = open session ▪ 01h = cancel session
P2	1	Reference control parameter P2	Session number
Lc	1	Length of the input data field	See data field in the command message

2.23.3.2 Data Field in the command message

Tag	Length	Value	Presence
85h	01h-14h	Public key ID	Conditional (when open session is requested, and when the key is searched by ID)
75h	01h – 3Ch	Public key label	Conditional (when open session is requested, and the key is searched by label)
A1h	01h	Mode of operation for the signature command	Conditional (when open session is requested)
91h	02h	Hash algorithm (See hash algorithms table)	Conditional (when open session is requested)
92h	01h	Signature algorithm (See signature algorithms table)	Conditional (when open session is requested)

2.23.4 Response Message

The response message returns SW1 and SW2 status word.

2.23.4.1 Processing state returned in response message

This command may either return a general error condition as listed in "General Error Conditions" or one of the following error conditions.

Status Word	Description	Possible cause
6700h	Wrong length (incorrect Lc)	Lc is not 00h (when close session is requested)
6985h	Conditions of use not satisfied	▪ Key reference not found ▪ Key not activated ▪ The public key doesn't have signature function permission ▪ Unsupported signature or hash algorithm ▪ Unsupported signature mode of operation
6989h	Maximum capacity reached	Maximum number of sessions reached
6A80h	Incorrect data	Issue in tags number or format
6A86h	Incorrect P1, P2	▪ P1: out of bound session request ▪ P2: session number incompatible with the session request (ex., request to open a session already opened, request to close a session for some other service)

2.24 Verify signature – update

2.24.1 Definition and scope

The **Verify Signature – Update** command is used to provide the applet with reference data to verify a signature and get the comparison result**.**

The command is used as many times as necessary to provide the applet with all reference data. According to the mode of operation for signature commands, reference data are as below:

- The full text or –exclusive- the final hash corresponding to it.

- The reference signature.

With the ‘full text’ mode several APDU might be necessary to provide reference data. In that case it is expected that all but the last data block are filled with 255 bytes of data.

Note: TLV must appear in the order they are listed.

2.24.2 Pre-Condition

The session is opened.

2.24.3 Command Message

2.24.3.1 Command header

Code	Length (byte)	Description	Possible value
CLA	1	Class	100000xxb or 1100xxxxb: no secure messaging (b7 to b1 encoded as in ISO7816-4 2013).
INS	1	Verify Signature – Update command	2Dh
P1	1	Reference control parameter P1	b8 b7 b6 b5 b4 b3 b2 b1 Meaning 0 - - - - - - - More incoming data 1 - - - - - - - Last incoming data - x x x x x x x RFU (must be 0)
P2	1	Reference control parameter P2	Session number
Lc	1	Length of the input data field	See data field in the command message

2.24.3.2 Data Field in the command message

Tag	Length	Value	Presence
9Bh	000000h - FFFFFFh	Data for which signature verification is requested	Conditional (when the full text is provided to the applet)
9Eh	01h	Final hash (See the hash algorithms constant table)	Conditional (when the final hash is provided to the applet)
33h	0040h – 0100h	Signature data, length from 40h for ECDSA 256 bits up to 0100h for RSA 2048 bits (See the algorithms for signature table)	Mandatory

2.24.4 Response Message

2.24.4.1 Processing state returned in response message

This command may either return a general error condition as listed in "General Error Conditions" or one of the following error conditions.

Status Word	Description	Possible cause
6700h	Wrong length (incorrect Lc)	Lc for a non-final block is not equal to 255
6D01h		Provided signature does not match
6985h	Conditions of use not satisfied	Length of the data to hash, length of the final hash, or length of the signature are not compatible with the requested operation
6A80h	Incorrect data	• Issue in tags number or format
6A86h	Incorrect P1, P2	▪ P1: the value differs from last incoming data or more incoming data ▪ P2: the session is not opened

3 Device-to-Applet Interface for IoT Security Applet Type 2

3.1 Summary of IoT Security Applet Type 2

The functional requirements for IoT Security Applet Type 2 are defined in document GSMA IoT.04 [1].

The applet is designed as a generic crypto toolbox application with specific (D)TLS (v1.2 and v1.3) related services.

The applet is based on a store holding objects, dedicated services to discover the capabilities of these objects, and services coming along.

The services exposed to the device are those listed in the SIM Alliance IoT work paper.

In the concept, the IoT security server (or the e/UICC issuer at personalization stage) is the one initializing the applet store: it creates necessary containers, sets object attributes, and when relevant set objects value. After this commissioning phase, the device can then request operations on objects present in the store, but in this version it cannot create any objects or reconfigure any.

The SCA (Symmetric Cryptography Applet) only supports services based on symmetric cryptography.

3.2 Overview of the applet store

The Applet Store has the following properties:

- It contains objects that are secret keys and files.
- An optional label (1 to 60 bytes) and an identifier (1 to 20 bytes) are given to each object (secret key and file). They are used as object reference in various services.
- Labels are associated to object containers while identifiers may be associated to containers or to the content.

  Note: It is assumed that label & ID policies set on the IoT security server are known on the device side. The device knows if IDs are used to identify containers or content and can thus use labels or IDs accordingly.

- Inside a type of object (secret key, files) object reference (identifiers, labels) must be unique. This rule does not apply to objects of different types.

Keys in the Applet Store have these attributes:

- Object access conditions: Specifies if the object (key) can be read or written from the device API.
- Object state: Specifies if the container holds an object (key, file) ready to use (by opposition to an empty or partially loaded key).
- Key type: only HMAC capable keys are supported
- Cryptographic functions: Specifies what cryptographic operation is allowed with the key. Only key derivation is supported.
- Algorithms for hash: Specifies what hash algorithm is allowed. Apply to key derivation where different hashes are accepted.
- Algorithms for key derivation [secret keys only]: Specifies what key derivation schemes are usable with the key.

Files in the Applet Store have these attributes:

- Object access conditions: Specifies if the object (file) can be read or written from the device API.
- Object state: Specifies if the container holds an object (file) ready to use (in contrast to an empty or partially loaded file).

It is assumed that the store might be updated by a remote administration server who can add, delete, or update objects from the applet store.

- In a situation where the execution of an operation requested by the remote administration server on a given applet store object is inconsistent with the possibility to execute a service requested by the device, this last request is denied, and the service requested by the device acknowledged with a dedicated SW telling that a remote administration operation is on-going.
- Upon remote administration operation completion, the next service requested by the device is executed normally but specific SW apply to let the device know that the applet store has changed.
- Note: In such situation, a device would presumably refresh its knowledge of the applet store using *Get Data* services.

3.3 Applet Services

The applet provides the following services:

- Discover what feature the applet can support. (Note: Notably because the applet can be extended with proprietary features).
- Discover what objects are there in the applet store and their attributes.

3.3.1 Applet Services to Run a (D)TLS1.2 Handshake

For the PSK-Plain scheme the applet provides:

- Get a random.

- Retrieve the PSK-Identity.
- Compute the master secret with the PRF function

3.3.2 Applet Services to Run a (D)TLS1.3 Handshake

For the PSK-Plain scheme the applet provides:

- Get a random.
- Retrieve the PSK-Identity.
- Compute the early secret with the HKDF function.
- (optional) Compute the handshake secret with the HKDF function.
- (optional) Compute the master secret with the HKDF function.

3.4 An Example of how to Set The Applet Store.

It is assumed that applet commissioning is from the IoT Security Server.

Note: Labels and identifiers are not illustrated

3.4.1 Example for the PSK-Plain Scheme

Client credentials, with an HMAC capable key for master secret PRF generation (TLS v1.2).

Key:

- Object access conditions (for keys): none.
- Object state (for keys): activated, value set by the IoT Security Server.
- Key type: HMAC.
- Cryptographic functions: key derivation.
- Algorithm for key derivation: PRF.

File, optional, to store identification data for the server:

- Object access conditions (for files): read.
- Object state (for files): activated (value downloaded).
- File specific usage: general purpose.

Note: The applet store can provide identification data (e/UICC identification and key identification data) for key lookup on the server side. This identification data will be transferred to the server in the ‘PSK identity’ field of the client KeyExchange message.

3.5 General data

3.5.1 Algorithms for hash

b16 – b9	b8	b7	b6	b5	b4	b3	b2	b1	Description
-	-	-	-	-	-	-	-	1	SHA-256
-	-	-	-	-	-	-	1	-	SHA-384 – optional
-	-	-	-	-	-	1	-	-	SHA-512 – optional
x	x	x	x	x	x				RFU, those bits must be 0

Note: HKDF can be used with different SHA algorithms

3.5.2 Algorithms for key derivation

b8	b7	b6	b5	b4	b3	b2	b1	Description
-	-	-	-	-	-	-	1	PRF SHA-256 (rfc5246)
-	-	-	-	-	-	1	-	HKDF (rfc5869)
x	x	X	x	x	x			RFU, those bits must be 0

3.5.3 Cryptographic functions

b8	b7	b6	b5	b4	b3	b2	b1	Description
-	-	-	-	-	-	-	1	Reserved
0	0	0	0	0	0	1	-	Reserved
0	0	0	0	0	1	-	-	Reserved
0	0	0	0	1	-	-	-	Key derivation
x	x	x	x					RFU, those bits must be 0

3.5.4 General status words

Status words listed below are applicable to all commands

Status Word	Description
63CFh	The command execution is successful but a remote administration session has completed.
6400h	integrity Issue (applet store integrity issue detected)
6984h	Referenced data invalidated (remote administration session on-going, including update on targeted applet store object)
6989h	Remote administration session including update on targeted applet store object has completed, and the command execution has failed
6A84h	Insufficient memory
6D00h	Invalid instruction
6E00h	Invalid class
6F00h	Unknown error (other error)
9000h	Successful execution

3.5.5 Hash algorithms constant

Hash	Length of message digest result
SHA-256	32 bytes
SHA-384	48 bytes
SHA-512	64 bytes

3.5.6 Key type

Key type value	Description
1 byte	
03h	Reserved
13h	Reserved
14h	Reserved

23h	Reserved
24h	Reserved
A0h	HMAC capable key
other	RFU

3.5.7 Object access conditions

b8	b7	b6	b5	b4	b3	b2	b1	Description
-	-	-	-	-	-	-	1	Read
-	-	-	-	-	-	1	-	Update
x	x	x	x	x	-	-	-	RFU, those bits must be 0

3.5.8 Object state

b8	b7	b6	b5	b4	b3	b2	b1	Description
-	-	-	-	-	-	-	1	1 = Activated (filled) 0 = Deactivated (container empty or partially filled)
x	x	x	x	x	x	x		RFU, those bits must be 0

3.6 Compute HKDF

3.6.1 Definition and scope

The Compute HKDF command is used to generate key material based on the HMAC-based key derivation function.

This command supports two modes of operation:

- HKDF-Extract general mode: the command is used to generate a pseudo random key. The input key material (IKM) comes as an input in command data.
- HKDF-Extract PSK-based: the command is used to generate a pseudo random key. The input key material (the secret) comes from a secret key of the applet store.

The hash function to use with the HKDF algorithm is indicated in input data.

Note: TLV must appear in the order they are listed.

3.6.2 Pre-Condition

The length of the salt must match with the length of the digest of the hash algorithm.

The secret key exists, is activated, it is granted with Key Derivation cryptographic function, and it is granted for HKDF key derivation algorithm (HKDF-Extract PSK-based).

3.6.3 Command Message

3.6.3.1 Command header

Code	Length (byte)	Description	Possible value
CLA	1	Class	100000xxb or 1100xxxxb: no secure messaging (b7 to b1 encoded as in ISO7816-4 2013)
INS	1	Compute HKDF command	4Ah
P1	1	Reference control parameter P1	HDKF generation mode • 00h: HKDF-extract general mode • 01h: HKDF-extract PSK-based
P2	1	Reference control parameter P2	00h
Lc	1	Length of the input data field	See data field in the command message
Le	1	Length of the response message	00h

3.6.3.2 Data Field in the command message

Tag	Length	Value	Presence
86h	01h-14h	Secret key ID	Conditional (when the key is searched by ID, And the HKDF-extract PSK-based generation mode is used)
76h	01h-3Ch	Secret key label	Conditional (when the key is searched by label, And the HKDF-extract PSK-based generation mode is used)
D1h	01h-40h	Secret	Conditional (when the HKDF-extract general mode is used)
D5h	20h-40h	Salt, length according to the digest of the hash function (See hash algorithms table)	Mandatory
91h	02h	Hash algorithm (See hash algorithms table)	Mandatory

3.6.4 Response Message

3.6.4.1 Data Field returned in response message

Name	Length	Value	Presence
Key material	20h – 40h	Pseudo-random value, length according to the digest of the hash function (See hash algorithms table)	Mandatory

3.6.4.2 Processing state returned in response message

This command may either return a general error condition as listed in "General Error Conditions" or one of the following error conditions.

Status Word	Description	Possible cause
6985h	Conditions of use not satisfied	• Secret key reference not found • The key is not activated • The secret key doesn't have key derivation cryptographic functions permission • The secret key is not granted with HKDF key derivation algorithm
6A80h	Incorrect data	• Issue in tags number or format
6A86h	Incorrect P1, P2	• P1: unknown generation mode • P2: must be 00h

3.7 Compute PRF

3.7.1 Definition and scope

The Compute PRF command generates pseudo-random numbers based on the PRF function.

This command supports two modes of operation:

- General mode: the secret comes as an input data in command data.
- PSK-plain pre-master secret mode: the applet builds a pre-master secret in conformance with rfc4279, chapter 2, and uses it as the secret in the PRF computation. The PSK used for that pre-master secret computation comes from a secret key of the applet store.

Note: TLV must appear in the order they are listed.

3.7.2 Pre-Condition

The key exists, is activated, it is granted with Key Derivation cryptographic function, and granted for PRF key derivation algorithm (PSK-plain pre-master secret mode).

3.7.3 Command Message

3.7.3.1 Command header

Code	Length (byte)	Description	Possible value
CLA	1	Class	100000xxb or 1100xxxxb: no secure messaging (b7 to b1 encoded as in ISO7816-4 2013)
INS	1	Compute PRF command	48h
P1	1	Reference control parameter P1	PRF generation mode • 00h: general mode • 01h: PSK-Plain pre-master secret mode
P2	1	Reference control parameter P2	00h
Lc	1	Length of the input data field	See data field in the command message
Le	1	Length of the response message	00h

3.7.3.2 Data Field in the command message

Tag	Length	Value	Presence
86h	01h-14h	Secret key ID	Conditional (when the key is searched by ID, and the PSK-Plain pre-master secret mode is used)
76h	01h-3Ch	Secret key label	Conditional (when the key is searched by label, and the PSK-Plain pre-master secret mode is used)
D1h	01h-40h	Secret	Conditional (when the general mode is used)
D2h	01h-7Bh	Label and Seed *(byte string, concatenation of the label and seed parameters)*	Mandatory
D3h	01h	Pseudo-random length	Mandatory

3.7.4 Response Message

3.7.4.1 Data Field returned in response message

Name	Length	Value	Presence
PRF output	Variable	Pseudo-random value	Mandatory

3.7.4.2 Processing state returned in response message

This command may either return a general error condition as listed in "General Error Conditions" or one of the following error conditions.

Status Word	Description	Possible cause
6985h	Conditions of use not satisfied	• Secret key reference not found • The key is not activated • The secret key doesn't have key derivation cryptographic functions permission • The secret key is not granted with PRF SHA-256 key derivation algorithm
6A80h	Incorrect data	• Issue in tags number or format
6A86h	Incorrect P1, P2	• P1: unknown generation mode • P2: must be 00h

3.8 Get data – application

3.8.1 Definition and scope

The **Get data - Application** command returns information about the applet and its capacity.

Note: TLV must appear in the order they are listed.

3.8.2 Pre-Condition

Void

3.8.3 Command Message

3.8.3.1 Command header

Code	Length (byte)	Description	Possible value
CLA	1	Class	100000xxb or 1100xxxxb: no secure messaging (b7 to b1 encoded as in ISO7816-4 2013)
INS	1	Get Data command	CBh
P1	1	Reference control parameter P1	00h = get data – application
P2	1	Reference control parameter P2	00h
Le	1	Length of the response message	35h - Expected application information

3.8.4 Response Message

3.8.4.1 Data Field returned in response message

Name	Length	Value			Presence
SIM Alliance version	03h bytes	**Tag**	**Length**	**Value**	Mandatory
		10h	01h	Version of the SIM Alliance applet specification	
Applet proprietary identifier	22h bytes	**Tag**	**Length**	**Value**	Mandatory
		11h	20h	32 byte proprietary applet identifier	
Max number of files	03h bytes	**Tag**	**Length**	**Value**	Mandatory
		B1h	01h	01h – FFh = maximum number of files	
Max number secret keys	03h bytes	**Tag**	**Length**	**Value**	Mandatory
		B4h	01h	01h – FFh = maximum number of secret keys	
Cryptographic functions	03h bytes	**Tag**	**Length**	**Value**	Mandatory
		90h	01h	Supported cryptographic functions coded as a combination of bits (See cryptographic function table)	
Supported algorithms for hash	04h bytes	**Tag**	**Length**	**Value**	Mandatory
		91h	02h	Supported algorithms for hash coded as a combination of bits (See algorithms for hash table)	
Supported algorithms for key derivation	03h bytes	**Tag**	**Length**	**Value**	Mandatory
		94h	01h	Supported algorithms for key derivation coded as a combination of bits (See algorithms for key derivation table)	

3.8.4.2 Processing state returned in response message

This command may either return a general error condition as listed in "General Error Conditions" or one of the following error conditions.

Status Word	Description	Possible cause
6700h	Wrong length (incorrect Le)	See Le possible value
6A86h	Incorrect P1, P2	• P1: unknown get data option • P2: different from 00h

3.9 Get data – file

3.9.1 Definition and scope

The **Get Data - File** command retrieves all information associated to a file in the applet store.

Note: TLV must appear in the order they are listed.

3.9.2 Pre-Condition

The targeted file must exist.

3.9.3 Command Message

3.9.3.1 Command header

Code	Length (byte)	Description	Possible value
CLA	1	Class	100000xxb or 1100xxxxb: no secure messaging (b7 to b1 encoded as in ISO7816-4 2013).
INS	1	Get Data command	CBh
P1	1	Reference control parameter P1	C3h = get data - file
P2	1	Reference control parameter P2	00h
Lc	1	Length of the input data field	See data field in the command message
Le	1	Length of the response message	00h

3.9.3.2 Data Field in the command message

Tag	Length	Value	Presence
73h	01h-3Ch	File label	Conditional (when the file is searched by label)
83h	01h-14h	File ID	Conditional (when the file is searched by ID)

3.9.4 Response Message

3.9.4.1 Data Field returned in response message

Name	Length	Value			Presence
		Tag	Length	Value	
File information structure	variable	C3h	variable	See file information structure in Get Data – Object List	Mandatory

3.9.4.2 Processing state returned in response message

This command may either return a general error condition as listed in "General Error Conditions" or one of the following error conditions.

Status Word	Description	Possible cause
6A80h	Incorrect data	Issue in tags number or format
6A82h	File not found	File reference not found
6A86h	Incorrect P1, P2	• P1: unknown get data option • P2: must be 00h

3.10 Get data – object list

3.10.1 Definition and scope

The **Get Data - Object List** command list all objects (files, secret keys) and their attributes present in the applet store.

Objects are listed one after the other. They are grouped together in the same response message. A response message only contains complete object information structures.

The ‘data outgoing mode’ (see P2) indicates where the first objects shall be fetched (first outgoing data) or where the next objects shall be fetched (after first outgoing data or more outgoing data). The presence of objects to retrieve is indicated with the SW 6300h. From that SW fetching subsequent objects requires next command to be provided with exact same CLA, INS and P1 bytes but P2.

Note: The order objects are returned is implementation dependent but TLV must appear in the order they are listed.

3.10.2 Pre-Condition

Void

3.10.3 Command Message

3.10.3.1 Command header

Code	Length (byte)	Description	Possible value
CLA	1	Class	100000xxb or 1100xxxxb: no secure messaging (b7 to b1 encoded as in ISO7816-4 2013).
INS	1	Get Data command	CBh
P1	1	Reference control parameter P1	01h = get data – object list
P2	1	Reference control parameter P2	Data outgoing mode • 00h = first outgoing data • 01h = more outgoing data
Le	1	Length of the response message	00h

3.10.4 Response Message

3.10.4.1 Data Field returned in response message

Name	Length	Value	Presence
File list	N * file attributes structure length	Tag: C3h; Length: variable; Value: File information structure	Conditional (as many as files in the applet store)
Secret key list	N * secret key information structure length	Tag: C4h; Length: variable; Value: Secret key information structure	Conditional (as many as secret key in the applet store)

3.10.4.2 File information structure

Name	Length	Value	Presence
File label	03h – 3Eh bytes	Tag: 73h; Length: 01h-3Ch; Value: File label	Conditional (if a label is associated to the file)
File ID	03h-16h bytes	Tag: 83h; Length: 01h-14h; Value: File ID	Mandatory
Object (file) access conditions	03h bytes	Tag: 60h; Length: 01h; Value: Objects access conditions coded as a combination of bits (See objects access conditions table)	Mandatory
Object state (for the file)	03h bytes	Tag: 4Ah; Length: 01h; Value: Object state (See object state table)	Mandatory
File size	04h bytes	Tag: 20h; Length: 02h; Value: File size	Mandatory

3.10.4.3 Secret key information structure

Name	Length	Value	Presence
Secret key label	03h – 3Eh bytes	Tag: 76h; Length: 01h – 3Ch; Value: Secret key Label	Conditional (if a label is associated to the secret key)
Secret key identifier	03h-16h bytes	Tag: 86h; Length: 01h-14h; Value: Secret key Id	Mandatory
Object access conditions (for the secret key)	03h bytes	Tag: 60h; Length: 01h; Value: Objects access conditions coded as a combination of bits (See object access conditions table)	Mandatory
Object state (for the secret key)	03h bytes	Tag: 4Ah; Length: 01h; Value: (See object state table)	Mandatory
Key type	03h bytes	Tag: 4Bh; Length: 01h; Value: (See key type table)	Mandatory
Supported algorithms for key derivation	03h bytes	Tag: 94h; Length: 01h; Value: Supported algorithms coded as a combination of bits (See algorithms for signature table)	Conditional (When the key is used for key derivation)

3.10.4.4 Processing state returned in response message

This command may either return a general error condition as listed in "General Error Conditions" or one of the following error conditions.

Status Word	Description	Possible cause
6300h	More data available	More data are available and can be retrieved with the more outgoing data mode
6A86h	Incorrect P1, P2	• P1: unknown get data option • P2: unknown data outgoing mode / CLA, INS, P1 bytes don't match with a more data outgoing mode

3.11 Get data – secret key information

3.11.1 Definition and scope

The **Get Data – Secret Key** command retrieves all information associated to a secret key.

Note: TLV must appear in the order they are listed.

3.11.2 Pre-Condition

The targeted key must exist.

3.11.3 Command Message

3.11.3.1 Command header

Code	Length (byte)	Description	Possible value
CLA	1	Class	100000xxb or 1100xxxxb: no secure messaging (b7 to b1 encoded as in ISO7816-4 2013).
INS	1	Get Data command	CBh
P1	1	Reference control parameter P1	C4h = get data – secret key
P2	1	Reference control parameter P2	00h
Lc	1	Length of the input data field	See data field in the command message
Le	1	Length of the response message	00h

3.11.3.2 Data Field in the command message

Tag	Length	Value	Presence
76h	01h-3Ch	Secret key label	Conditional (when the key is searched by label)
86h	01h-14h	Secret key ID	Conditional (when the key pair is searched by ID)

3.11.4 Response Message

3.11.4.1 Data Field returned in response message

Name	Length	Value			Presence
		Tag	Length	Value	
Secret key information structure	variable	C4h	variable	See secret key information structure in Get Data – Object List	Mandatory

3.11.4.2 Processing state returned in response message

This command may either return a general error condition as listed in "General Error Conditions" or one of the following error conditions.

Status Word	Description	Possible cause
6985h	Conditions of use not satisfied	Key reference not found
6A80h	Incorrect data	Issue in tags number or format
6A86h	Incorrect P1, P2	• P1: unknown get data option • P2: must be 00h

3.12 Get random

3.12.1 Definition and scope

Get Random command returns a random value. Its length can extend from 1 byte up to 256 bytes.

3.12.2 Pre-Condition

Void.

3.12.3 Command Message

3.12.3.1 Command header

Code	Length (byte)	Description	Possible value
CLA	1	Class	100000xxb or 1100xxxxb: no secure messaging (b7 to b1 encoded as in ISO7816-4 2013).
INS	1	Get random command	84h
P1	1	Reference control parameter P1	00h
P2	1	Reference control parameter P2	00h
Le	1	Length of the response message	Variable, see the response message (00h means 256 bytes expected)

3.12.4 Response Message

3.12.4.1 Data Field returned in response message

Name	Length	Value	Presence
Card random	variable	Random value	Mandatory

3.12.4.2 Processing state returned in response message

This command may either return a general error condition as listed in "General Error Conditions" or one of the following error conditions.

Status Word	Description	Possible cause
6A86h	Incorrect P1, P2	P1, P2 are not equal to 00h

3.13 Read file

3.13.1 Definition and scope

The **Read File** command reads the contents of an EF from the start offset and returns 256 bytes or less if it reaches the end file

Note: EF are all located under the MF (there is no subdirectory).

Note: TLV must appear in the order they are listed.

3.13.2 Pre-Condition

The targeted file must exist, must be activated, and must be granted with read object access condition.

3.13.3 Command Message

3.13.3.1 Command header

Code	Length (byte)	Description	Possible value
CLA	1	Class	100000xxb or 1100xxxxb: no secure messaging (b7 to b1 encoded as in ISO7816-4 2013).
INS	1	Read File command	B0h
P1	1	Reference control parameter P1	Offset in the file
P2	1	Reference control parameter P2	
Lc	1	Length of the input data field	See data field in the command message
Le	1	Length of the response message	00h

3.13.3.2 Data Field in the command message

Tag	Length	Value	Presence
83h	01h-14h	File ID	Conditional (when the file is selected by identifier)
73h	01h – 3Ch	File label	Conditional (when the file is selected by label)

3.13.4 Response Message

3.13.4.1 Data Field returned in response message

Name	Length	Value	Presence
File content	variable	Data read	Mandatory

3.13.4.2 Processing state returned in response message

This command may either return a general error condition as listed in "General Error Conditions" or one of the following error conditions.

Status Word	Description	Possible cause
6981h	Command incompatible with file structure	Given offset is outside EF limits
6985h	Conditions of use not satisfied	• Read is not granted • The file is deactivated
6A80h	Incorrect data	Issue in tags number or format
6A82h	File not found	File reference not found

Annex A Document Management

A.1 Document History

Version	Date	Brief Description of Change	Approval Authority	Editor / Company
1.0	6th Nov 2019	New document.	Technology Group	Ian Smith / GSMA

A.2 Other Information

Type	Description
Document Owner	IoT Programme
Editor / Company	Ian Smith / GSMA

It is our intention to provide a quality product for your use. If you find any errors or omissions, please contact us with your comments. You may notify us at prd@gsma.com

Your comments or suggestions & questions are always welcome.