G+D Mobile Security

SIGNiT Application

Customer Manual

G+D Mobile Security GmbH
January 2020

Content

Document version history

Date	Version	Modification
19/08/2019	1.0	Initial Version
13/09/2019	2.0	Corrections in description of SIGN command for P1 parameter
08/10/2019	3.0	Link to Ubirch protocol description included Description of Verify PIN command included
24/01/2020	4.0	Informal information and description of Put Key command added Hash (sha256) payload before building the Ubirch package in Signing operation included in table "Sign Init P1 Reference Parameter Codification"

1. Introduction

The purpose of this document is to describe the commands of the SIGNiT application and its functionality. It is a java card compatible smartcard application which can be loaded to secure elements or standard SIM cards. It is used to a act as a secure storage of confidential key material and corresponding certificates which is the base to calculate digital signatures and verification of it.

1.1 Functional Overview

The SIGNiT application provides APDU command interface to the hosting device for adding, updating and deleting Public and private keys and certificates. Each key entry is a container that consists of an ID and a Title (optionally). The Title identifies the entry with a user readable text whereas the ID is used to reference the entry within the secure storage.

In addition it provides a APDU command set for cryptographic operations which are

- Signature Calculation
- Signature Verification
- Secure Random Generation
- Key pair Generation
- Certificate Sign Request Generation

In order to carry out any cryptographic operation the application needs to be personalised with the corresponding key's (i.e. RSA, ECC) and certificates (i.e. X-509). Once the key or certificate is stored in the SIGNiT application an credential identifier will be returned to the off-card entity which is used to address further operations.

1.2 Overview APDU commands

The list of commands supported by the SIGNiT Application are listed in table below. In further sections, these commands will be explained more in detail.

Command Type	Command Name
General commands	Select Application
	Generate Secure Random
	Select SS Entry
	Delete All SS Entries
	Generate Key Pair
	Generate CSR
	Get Key
	Store Certificate
	Update Certificate
	Get Certificate
Cryptographic commands	Sign Init
	Sign Update/Final
	Verify Signature Init
	Verify Signature Update/Final

Table 1. List of Supported Commands

2. APDU Commands

2.1 Secure Storage Commands

2.1.1 Select Application

Select Application APDU is intended to select the SIM card based application for direct communication and operations requests.

The Create SS Entry APDU format of the command is described as follows:

Command	CLA	INS	P1	P2	Lc	Data
Select SIGNiT application	'00'	'A4'	'04'	'00'	'10'	D2 76 00 01 18 00 02 FF 34 10 83 89 C0 02 8B 02

Table 2. Select application APDU format

Status conditions returned by the applet:

SW	Reason
'90 00'	Command performed successfully.
'6A 86'	Incorrect parameter P1 or P2.
'69 82'	Security status not satisfied

Table 3. Select application response codes

2.1.2 Verify PIN

The Verify PIN command APDU format of the command is described as follows:

Command	CLA	INS	P1	P2	Lc	Data
Verify PIN	'00'	'20'	'00'	'00'	Var.	See table Verify PIN data command

Table 4. Verify PIN Command APDU format

Verify PIN command data shall be encoded according to:

Name	Length	Value	Presence
PIN value	Var.	'xxxx..'	Mandatory

Table 5. Verify PIN Command Data

NOTES:

- If only one PIN try is remaining and after another false PIN presentation, the respective PIN will be blocked and applet will return SW '63C0' for indicating that PIN has been just blocked.

Status conditions returned by the applet:

SW	Reason
'90 00'	Command performed successfully.
'6A 86'	Incorrect parameter P1 or P2.
'67 00'	PIN length out of range.
'69 83'	PIN already blocked.
'63 Cx'	Wrong PIN value, x tries left.

Table 6. Verify PIN Status Conditions

2.1.3 Generate Secure Random

Generate Secure Random command is intended to retrieve a secure random number generated by the secure element.

The Generate Secure Random command APDU format of the command is described as follows:

Command	CLA	INS	P1	P2	Le

Generate Secure Random	'80'	'B9'	'00'	'XX'	'00'

Table 7. Generate Secure Random Command APDU format

P2: indicates the length in bytes of the Generate Secure Random number to be generated. Length could be from 1 byte to 254 bytes.

Generate Secure Random Response shall be encoded according to:

Name	Length	Value	Presence
Secure Random Number Generated	P2	'xxxx..'	Mandatory

Table 8. Generate Secure Random Response Data

Status conditions returned by the applet for Generate Secure Random:

SW	Reason
'90 00'	Command performed successfully.
'6A 86'	Incorrect parameter P1
	Length indicated in P2 out of range
'67 00'	Data field not empty

Table 9. Generate Secure Random Status Conditions

2.1.4 Select SS Entry

Select SS Entry command must be used to select an entry from the Secure Storage.

The Select SS Entry APDU format of the command is described as follows:

Command	CLA	INS	P1	P2	Lc	Data
Select SS Entry	'80'	'A5'	See table: SS Entry P1 Reference Parameter	'00'	Var.	See table: Select SS Entry Command Data

Table 10. Select SS Entry Command APDU format

P1 byte shall be coded according to the following table:

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
-	-	-	-	-	-	0	0	Select by ID. Select the entry referenced by the ID
-	-	-	-	-	-	0	1	Select First. Select the First SS Entry
-	-	-	-	-	-	1	0	Select Next. Select next available SS Entry
X	X	X	X	X	X	-	-	RFU (those bits shall be 0)

Table 11. Select SS Entry P1 Reference Parameter Codification

Select SS Entry Command Data shall be encoded according to:

Name	Length	Value	Presence

Identifier of SS Entry to be selected	Var.	'xxxx…'	Conditional

Table 12.Select SS Entry Command Data

Select SS Entry Response shall be encoded according to:

TAG	LEN	Value Description	Value	Presence
'C4'	Var.	The ID of the referenced SS Entry	'xxxx…'	Mandatory
'C0'	Var.	The title of the referenced SS Entry	'xxxx…'	Mandatory

Table 13.Select SS Entry Response Data

Status conditions returned by the applet:

SW	Reason
'90 00'	Command performed successfully.
'6A 86'	Incorrect parameter P1 or P2.
'6A 80'	Wrong Data introduced.
'6A 88'	Entry not found.
	Last Entry is selected and Select Next SS Entry command is received.
	Select Next SS Entry received in the middle of Create SS Entry Flow.
'69 82'	Security status not satisfied

Table 14.Select SS Entry Status Conditions

2.1.5 Delete SS Entry

Delete SS Entry command is intended to delete any type of entry from the secure storage.

The Delete SS Entry APDU format of the command is described as follows:

Command	CLA	INS	P1	P2	Lc	Data
Delete SS Entry	'80'	'E4'	'00'	'00'	Var.	See table: Delete SS Entry Command Data

Table 15. Delete SS Entry Command APDU format

Name	Length	Value	Presence
Entry ID to be deleted	Var.	'xxxx..'	Mandatory

Table 16. Delete SS Entry Command Data

Status conditions returned by the applet:

SW	Reason
'90 00'	Command performed successfully.
'6A 86'	Incorrect parameter P1 or P2.
'6A 80'	Entry ID length out of range
'6A 88'	Entry not found.
'69 82'	Security status not satisfied

Table 17. Delete SS Entry Status Conditions

2.1.6 Put Key

Put Key command is intended to store a key in the secure storage.

The Put Key APDU format of the command is described as follows:

Command	CLA	INS	P1	P2	Lc	Data
Put Key	'80'	'D8'	See table Put Key P1 Reference Parameter codification	'00'	Var.	See table Put Key command data

Table 18. Put Key Command APDU format

NOTES:

- This command is addressed to the applet and requires PIN verification towards the application.
- After successful PUT KEY command, the newly created secure storage entry is selected.

P1 byte shall be codified according to the following table:

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
1	-	-	-	-	-	-	-	Last (or only) command
0	-	-	-	-	-	-	-	More PUT KEY commands
-	X	X	X	X	X	X	X	RFU (those bits shall be 0)

Table 19. Put Key P1 Reference Parameter Codification

The Command data should be according to the following table:

TAG	LEN	Value Description	Value	Presence
'C4'	Var.	Entry ID	'xxxx…'	Mandatory
'C0'	Var.	Entry Title	'xxxx…'	Optional
'C1'	1	Permissions	See table Key Permission Codification	Optional
'C2'	3	Key Type + Key Length	See table Put Key values and length	Mandatory
'C3' / 'E3'	Var.	Plain Key Component Value / Encrypted Key Component	'xxxx…' / See table Put Key – Encryption Key Component	Mandatory
'C3' / 'E3'	Var.	Plain Key Component Value / Encrypted Key Component	'xxxx…' / See table Put Key – Encryption Key Component	Conditional

Table 20. Put Key Command Data

TAG	LEN	Value Description	Value	Presence
'D0'	1	Algorithm type	See table Put Key Encryption Supported Algorithms	Mandatory
'D1'	Var.	Key ID of Encryption Key	'00 01' - MAX	Mandatory
'D2'	Var.	Encrypted Key Component Value	'xxxx…'	Mandatory

Table 21. Put Key – Encryption Key Component

NOTES:

- If Entry title tag is not present or its length is equal to zero, the entry is created with an empty title.
- Permissions shall be coded according to the following table:

Value	Meaning
'00'	No Permissions
'01'	Only Read Allowed
'02'	Only Write Allowed
'03'	Read & Write Allowed

Table 22. Put Key Permissions Codification

- If Permissions tag is not present, key will be stored using permissions default value for keys ('02' → Only Write allowed)

- Key Type value are reflected in the following table:

Value	Meaning
'0F'	TYPE_AES
'15'	TYPE_HMAC
'05'	TYPE_RSA_PRIVATE
'04'	TYPE_RSA_PUBLIC
'0C'	TYPE_EC_FP_PRIVATE
'0B'	TYPE_EC_FP_PUBLIC

Table 23. Put Key - Key Type Values

- Key Length values shall be introduced in bits according to the following table:

Value	Meaning
'00 80'	LENGTH_AES_128
'01 00'	LENGTH_AES_256
'02 00'	LENGTH_HMAC_512
'08 00'	LENGTH_RSA_2048
'01 00'	LENGTH_EC_FP_256

Table 24. Put Key – Key Length Values

- Key components allowed to be put are the following ones:
  - RSA Public Key Exponent
  - RSA Public Key Modulus
  - RSA Private Key Exponent
  - RSA Private Key Modulus
  - ECC Public Key
  - ECC Private Key
  - AES Symmetric Key
  - HMAC Symmetric Key
- It is possible to store a key component encrypted with a symmetric key and fulfilling strength restrictions indicated in the Table "Encryption Key Strength" below, taking into account the current supported symmetric keys. In that case, this symmetric key shall already exist inside Secure Storage. For introducing encrypted key components, 'E3' tag shall be used instead and it shall be composed as indicated in table "Put Key – Encryption Key Component".

Key to be stored	Acceptable Strength Encryption Key
AES128	AES128 or AES256
AES256	AES256
HMAC	
RSA	
ECC	

Table 25. Encryption Key Strength

- In case of storing an already encrypted key, algorithm to be used shall be coded as in the following table:

Value	Meaning
'1A'	ALG_AES_ECB_ISO9797_M2

Table 26. Put Key Encryption Supported Algorithms

- When storing any asymmetric key (either RSA or ECC), private and public components shall be stored in a different Secure Storage entry using different PUT KEY sequence commands.
- When storing a RSA key, modulus and exponent shall be included in the same sequence of PUT KEY commands. It shall be done using two Key Component tags (any of them can be either plain or encrypted). In that case, Key Exponent shall be **always** introduced before Key Modulus. If Key Modulus is introduced before Key Exponent, unexpected behaviour is expected.
- Sensitive data shall be sent using chaining mechanism (More Put Key) in case data size is greater than maximum APDU command data field size (255 bytes).

Put Key Response shall be empty.

Status conditions returned by the applet for Put Key Command:

SW	Reason
'90 00'	Command performed successfully.
'6A 86'	Incorrect parameter P1 or P2.
'69 82'	Security status not satisfied
	PIN Verification Enabled and PIN not verified
'69 83'	PIN blocked
'6A 80'	Invalid Tag
	Mandatory Tag missing
	Key Tag and Key Length not consistent
	Key Type or Algorithm not supported
	More than one Key Component Indicated (unless RSA Key is being stored)
	RSA Exponent component is indicated, but RSA Modulus not
	Encryption Key Strength not enough
	Either Entry ID or Entry Title already exists.
	Entry ID or Title length out of range
'6A 88'	Encryption Key not found
'6A 84'	No memory available for creating a new entry

Table 27. Put Key Status Conditions

2.1.7 Generate Key Pair

Generate Key Pair command is intended to generate and store or replace a public-private key pair into the secure storage.

The Generate Key Pair APDU format of the command is described as follows:

Command	CLA	INS	P1	P2	Lc	Data
Generate Key Pair	'80'	'B2'	See table: Generate Key Pair P1 Reference Parameter Codification	'00'	Var.	See table: Generate Key Pair Command Data

Table 28. Generate Key Pair command APDU format

P1 byte shall be codified according to the following table:

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
1	-	-	-	-	-	-	-	Generate ECC NIST P-256 Key Pair As specified in Federal Information Processing Standards Publication 186-4: Digital Signature Standard (DSS).
-	X	X	X	X	X	X	X	RFU (those bits shall be 0)

Table 29. Generate Key Pair P1 Reference Parameter Codification

The Command data should be according to the following table:

TAG	LEN	Value Description	Value	Presence
'C4'	Var.	Entry ID of the Public Key	'xxxx…'	Mandatory
'C0'	Var.	Entry Title for Public Key	'xxxx…'	Optional
'C1'	1	Permissions for Public Key	See table: Key Permissions Codification	Optional
'C4'	Var.	Entry ID of the Private Key	'xxxx…'	Mandatory
'C0'	Var.	Entry Title for Private Key	'xxxx…'	Optional
'C1'	1	Permissions for Private Key	See table: Key Permissions Codification	Optional

Table 30. Generate Key Pair Command Data

Status conditions returned by the applet for Generate Key Pair Command:

SW	Reason
'90 00'	Command performed successfully.
'6A 86'	Incorrect parameter P1 or P2.
'6A 80'	Invalid Tag
	Mandatory Tag missing
	New key algorithm different from key to be replaced algorithm
	Title length out of range
	Entry ID indicated for both Public Key and Private Key is the same
'6A 88'	Entry ID tried to be replaced is not a key
'69 86'	Write permission denied

'6A 84'	No memory available for creating a new entry
'69 82'	Security status not satisfied

Table 31. Generate Key Pair Status Conditions

2.1.8 Generate Certificate Sign Request

Generate Certificate Sign Request command is intended to obtain a Certificate Sign Request to be sent to a Certification Authority.

The Generate Certificate Sign Request command APDU format of the command is described as follows:

Command	CLA	INS	P1	P2	Le
Generate CSR	'80'	'BA'	See table P1 coding of CSR command	'00'	See table CSR command data coding

Table 32. Generate CSR command APDU format.

Reference control parameter P1 shall be coded according to the following table:

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
1	-	-	-	-	-	-	-	Last block
0	-	-	-	-	-	-	-	More blocks
-	-	-	-	-	-	-	0	Get First (or only) data part
-	-	-	-	-	-	-	1	Get next data part
-	X	X	X	X	X	X	-	RFU (those bits shall be 0)

Table 33. P1 coding of CSR command.

The CSR command data should be encoded according to the following table:

Tag	Length	Value Description	Presence
'C4'	Var.	Public Key ID of the key to be used as the Public Key carried in the CSR	Mandatory
'C4'	Var.	Private Key ID of the key to be used for signing the CSR	Mandatory
'E5'	Var.	Certification Request parameters (see table below)	Mandatory

Table 34. CSR command data coding.

TAG	LEN	Value Description	Value	Presence
'D3'	1	Version	'00', '01', '02',…	Mandatory
'E7'	Var.	Subject Information (Relative Distinguished Names)	See table Distinguish names parameters	Mandatory
'C2'	3	Subject PKI Algorithm Identifier (Key Type + Key Length)	'0B 01 00' (TYPE_EC_FP_PUBLIC + LENGTH_EC_FP_256)	Mandatory
'E9'	Var.	CRI Attributes	Optional PKCS#9 attributes	Optional
'D0'	1	Signature Algorithm Identifier	'21' (ALG_ECDSA_SHA_256)	Mandatory

Table 35. Certification Request parameters

TAG	LEN	Value Description	Format	Example	Presence
'D4'	Var.	countryName	ASCII	DE: '44 45'	Optional*
'D5'	Var.	stateOrProvinceName	UTF-8 String	Bayern: '42 61 79 65 72 6E'	Optional*
'D6'	Var.	localityName	UTF-8 String	Munich: '4D 75 6E 69 63 68'	Optional*
'D7'	Var.	organizationName	UTF-8 String	G&D: '47 26 44'	Optional*
'D8'	Var.	organizationalUnitName	UTF-8 String	GDM: '47 44 4D'	Optional*
'D9'	Var.	commonName	UTF-8 String	gd.com: '67 64 2E 63 6F 6D'	Optional*
'DA'	Var.	emailAddress	ASCII	gd@gd.com: '67 64 40 67 64 2E 63 6F 6D'	Optional

Table 36. Distinguished names parameters.

* It's Mandatory to set at least one of the Distinguished Names parameters (from D4 to D9).

Notes:

- The only algorithm supported for CRS generation is ECDSAwithSHA256 (ALG_ECDSA_SHA_256).
- To configure the CRI Parameters (Optional PKCS#9 Attributes) is required to set them in ASN.1 format, exactly as it's supposed to appear in the CSR command generated.

Status conditions returned by the applet for Generate CSR command:

SW	Reason
'90 00'	Command performed successfully.
'63 10'	Command performed successfully and More data available. Get next data part required.
'6A 86'	Incorrect parameter P1 or P2.
'67 00'	Data field contains data when P1 indicates Get Next Data Part.
'6A 88'	Public Key or Private Key not found.
'6A 80'	Wrong data. - Mandatory field not present. - Unknown tag. - Generic error in data format. - Signature Algorithm not supported. - Key type inconsistent.
'69 85'	Conditions of use not satisfied.
'69 82'	PIN Verification Enabled and PIN not verified.
'69 83'	PIN blocked.

Table 37. Generate CSR Conditions.

2.1.9 Get-Read Key

Get Key command is intended to retrieve an existing key in the secure storage.

The Get Key APDU format of the command is described as follows:

Command	CLA	INS	P1	P2	Lc	Data

Get Key	'80'	'CB'	See table: Get Key P1 Reference Parameter	'00'	Var.	See table: Get Key Response Data

Table 38. Get Key Command APDU format

P1 byte shall be codified according to the following table:

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
-	-	-	-	-	-	-	0	Get first (or only) data part
-	-	-	-	-	-	-	1	Get next data part
X	X	X	X	X	X	X	X	RFU (those bits shall be 0)

Table 39. Get Key P1 Reference Parameter Codification

NOTES:

- An SS Entry shall be previously selected in order to carry out Get Key command properly.

Get Key Command Data shall be encoded according to:

TAG	LEN	Value Description	Value	Presence
'D0'	1	Algorithm type	See table: Get Key Algorithm Type	Mandatory
'D1'	Var.	Key ID of Encryption Key	'xxxx…'	Conditional

Table 40. Get Key Command Data

Value Description	Value
Without encryption	'00'
ALG_AES_ECB_ISO9797_M2	'1A'

Table 41. Get Key Algorithm Type

Get Key Response shall be encoded according to:

TAG	LEN	Value Description	Value	Presence
'C2'	3	Key Type + Key Length	See table: Key – Key Type Values Key – Key Length Values	Mandatory
'C3'	Var.	Plain Key Component Value	'xxxx…'	Conditional
'E3'	Var.	Encrypted Key Component	See table: Key – Encryption Key Component	Conditional

Table 42. Get Key Response Data

Status conditions returned by the applet for Get Key Command:

SW	Reason
'90 00'	Command performed successfully. No more data available
'63 10'	More data available. Get next data part required.

‘6A 86’	Incorrect parameter P1 or P2.
’67 00’	Get Next data part indicated and data field not empty
’69 85’	Get Next data part indicated before Get First data part
‘6A 80’	Mandatory tag missing
	Invalid Tag
	Encryption Key Strength not enough
‘6A 88’	No Key Entry is selected
	Encryption Key not found
’69 86’	Read Permission Denied
’69 82’	Security status not satisfied

Table 43. Get Key Status Conditions

2.1.10 Store Certificate

Store Certificate command is intended to store certificates into the secure storage which will be used for performing crypto operations.

The Store Certificate APDU format of the command is described as follows:

Command	CLA	INS	P1	P2	Lc	Data
Store Certificate	‘80’	‘E3’	See table: Store Certificate P1 Reference Parameter Codification	‘00’	Var.	See table: Store Certificate Command Data

Table 44. Store Certificate Command APDU Format

P1 byte shall be codified according to the following table:

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
0	-	-	-	-	-	-	-	More blocks
1	-	-	-	-	-	-	-	Last (or only) block
-	X	X	X	X	X	X	X	RFU (those bits shall be 0)

Table 45. Store Certificate P1 Reference Parameter Codification

The Command data should be according to the following table:

TAG	LEN	Value Description	Value	Presence
‘C4’	Var.	Entry ID	‘xxxx…’	Mandatory
‘C0’	Var.	Entry Title	‘xxxx…’	Optional
‘C1’	1	Permissions	See table: Key Permissions Codification	Optional
‘C3’	Var.	Certificate Value	‘xxxx…’	Mandatory

Table 46. Store Certificate Command Data

Status conditions returned by the applet for Store Certificate Command:

SW	Reason

’90 00’	Command performed successfully.
‘6A 86’	Incorrect parameter P1 or P2.
‘6A 80’	Invalid Tag
	Mandatory Tag missing
	Certificate algorithm not supported
	Invalid Certificate
	Entry ID already exists
	Entry ID or Title length out of range
‘6A 84’	No memory available for creating a new entry
’69 82’	Security status not satisfied

Table 47. Store Certificate Error Conditions

2.1.11 Update Certificate

Update Certificate command is intended to modify an existing certificate in the secure storage.

The Update Certificate APDU format of the command is described as follows:

Command	CLA	INS	P1	P2	Lc	Data
Update Certificate	‘80’	‘E6’	See table: Update Certificate P1 Reference Parameter Codification	‘00’	Var.	See table: Update Certificate Command Data

Table 48. Update Certificate Command APDU Format

P1 byte shall be codified according to the following table:

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
1	-	-	-	-	-	-	-	Last (or only) command
0	-	-	-	-	-	-	-	More UPDATE CERTIFICATE commands
-	X	X	X	X	X	X	X	RFU (those bits shall be 0)

Table 49. Update Certificate P1 Reference Parameter Codification

The Command data should be according to the following table:

Tag	Length	Value Description	Value	Presence
‘C4’	Var.	Entry ID	‘xxxx…’	Optional
‘C0’	Var.	Entry Title	‘xxxx…’	Optional
‘C1’	1	Permissions	See table: Key Permissions Codification	Optional
‘C3’	Var.	Certificate Value	‘xxxx…’	Mandatory

Table 50. Update Certificate Command Data

Response Data shall be empty.

Status conditions returned by the applet for Update Certificate Command:

SW	Reason
’90 00’	Command performed successfully.
‘6A 86’	Incorrect parameter P1 or P2.
’69 82’	Security status not satisfied
‘6A 88’	No Certificate Entry is Selected
‘6A 80’	Invalid Tag
	Mandatory Tag missing
	Wrong Certificate algorithm
	Invalid Certificate
’69 86’	Write Permission Denied

Table 51. Update Certificate Status Conditions

2.1.12 Get Certificate

Get Certificate command is intended to retrieve an existing certificate in the secure storage.

The Get Certificate APDU format of the command is described as follows:

Command	CLA	INS	P1	P2	Le
Get Certificate	‘80’	‘CC’	See table: Get Certificate P1 Reference Parameter Codification	‘00’	’XX’

Table 52. Get Certificate command APDU format

P1 byte shall be codified according to the following table:

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
-	-	-	-	-	-	-	0	Get first (or only) data part
-	-	-	-	-	-	-	1	Get next data part
X	X	X	X	X	X	X	X	RFU (those bits shall be 0)

Table 53. Get Certificate P1 Reference Parameter Codification

Get Certificate Response shall be encoded according to:

Tag	Length	Value Description	Value	Presence
‘C3’	Var.	Certificate Value	‘xxxx…’	Mandatory

Table 54. Get Certificate Response Data

Status conditions returned by the applet for Get Certificate Command:

SW	Reason
’90 00’	Command performed successfully. No more data available
’63 10’	More data available. Get next data part required.
‘6A 86’	Incorrect parameter P1 or P2.

’67 00’	Data field present
’69 85’	Get Next data part indicated before Get First data part
‘6A 88’	No Certificate Entry is Selected
’69 86’	Read Permission Denied
	Not Certificate Entry Selected
’69 82’	Security status not satisfied

Table 55. Get Certificate Status Conditions

2.2 Cryptographic Commands

2.2.1 Sign Initialize

Sign Init command is intended to intended to initialise a signature operation.

The Sign command APDU format of the command is described as follows:

Command	CLA	INS	P1	P2	Lc	Data
Sign Init	‘80’	‘B5’	See table: Verify Signature Init P1 Reference Parameter Codification	See table: Sign Init P2 Reference Parameter Codification	Var.	See table: Sign Init Command Data

Table 56. Sign Command APDU format

Reference control parameter P1 indicates the Data Format and shall be coded according to the following table:

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
0	0	0	0	0	0	0	0	‘Regular’ Signing.
0	0	1	0	0	0	1	0	Ubirch Protocol Version 2 – Simple Message
0	0	1	0	0	0	1	1	Ubirch Protocol Version 2 – Chained Message
-	0	-	0	0	0	-	-	Received Payload is directly (without hashing) used to build the Ubirch Package
-	1	-	0	0	0	-	-	Received Payload is hashed before it is used to build the Ubirch Package
X	-	-	X	X	X	-	-	RFU (those bits shall be 0)

Table 57: Sign Init P1 Reference Parameter Codification.

Details of the Ubirch protocol are described at the following link:
https://developer.ubirch.com/utp.html#how-it-works-in-a-nutshell

Reference control parameter P2 shall indicate the Data Format and shall be coded according to the following table:

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
-	-	-	-	-	-	-	0	Entry ID + Algorithm

-	-	-	-	-	-	-	1	RFU
X	X	X	X	X	X	X	-	RFU (those bits shall be 0)

Table 58. Sign Init P2 Reference Parameter Codification

The Sign command data should be encoded according to the following table:

TAG	LEN	Value Description	Value	Presence
'C4'	Var.	Entry ID of Sign Key	'xxxx…'	Mandatory
'D0'	1	Algorithm to be used	See table: Sign Init Supported Algorithms	Mandatory

Table 59. Sign Init Command Data

Value	Meaning
'19'	ALG_HMAC_SHA_256
'1B'	ALG_HMAC_SHA_512
'21'	ALG_ECDSA_SHA_256

Table 60. Sign Init Supported Algorithms

Status conditions returned by the applet for Sign Init:

SW	Reason
'90 00'	Command performed successfully.
'6A 86'	Incorrect parameter P1 or P2.
'6A 80'	Algorithm not supported.
	Entry ID and algorithm inconsistent
	Invalid tag
	Mandatory tag missing
'6A 88'	Entry not found
'69 82'	Security status not satisfied

Table 61. Sign Init Status Conditions

2.2.2 Sign Update/Final

Sign Update/Final command is intended to perform a signature of the given payload using the information provided in a previous Sign Init command.

The Sign Update/Final command APDU format of the command is described as follows:

Command	CLA	INS	P1	P2	Lc	Data
Sign Update/Final	'80'	'B6'	See table: Sign Update/Final P1 Reference Parameter Codification	'00'	Var.	See table: Sign Update/Final Command Data

Table 62. Sign Update/Final Command APDU format

Reference control parameter P1 shall be coded according to the following table:

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
1	-	-	-	-	-	-	-	Last block (Final Sign)
0	-	-	-	-	-	-	-	More blocks (Update Sign)
-	-	-	-	-	-	-	0	Get first (or only) data part
-	-	-	-	-	-	-	1	Get next data part
-	X	X	X	X	X	X	-	RFU (those bits shall be 0)

Table 63. Sign Update/Final P1 Reference Parameter Codification

The Sign command data should be encoded according to the following table:

Length	Value Description	Presence
Var.	Data to be signed	Mandatory / Conditional

Table 64. Sign Update/Final Command Data

Sign Response shall be encoded according to:

Name	Length	Value	Presence
Signed Data	Var.	'xxxx..'	Mandatory

Table 65. Sign Update/Final Response Data

NOTES:

- In case of using ALG_ECDSA_SHA_256 algorithm, the signed data shall be encoded as an ASN.1 sequence of two INTEGER values, r and s.

Status conditions returned by the applet for Sign Update/Final:

SW	Reason
'90 00'	Command performed successfully.
'63 10'	Command performed successfully and More data available. Get next data part required.
'6A 86'	Incorrect parameter P1 or P2.
'67 00'	Data field contains data when P1 indicates Get Next Data Part
'69 85'	Sign Init command not previously sent.
	Sign Final Next when there are no data to be retrieved

Table 66. Sign Update/Final Status Conditions

2.2.3 Verify Signature Initialize

Verify Signature Init command is intended to initialise a signature verification operation.

The Verify Signature command APDU format of the command is described as follows:

Command	CLA	INS	P1	P2	Lc	Data

Verify Signature Init	'80'	'B7'	See table: Verify Signature Init P1 Reference Parameter Codification	See table: Verify Signature Init P2 Reference Parameter Codification	Var.	See table: Verify Signature Init Command Data

Table 67. Verify Signature Init Command APDU format

Reference control parameter P1 indicates the Data Format and shall be coded according to the following table:

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
0	0	0	0	0	0	0	0	'Regular' Signing.
0	0	1	0	0	0	1	0	Ubirch Protocol Version 2 – Simple Message
0	0	1	0	0	0	1	1	Ubirch Protocol Version 2 – Chained Message
X	X	-	X	X	X	-	-	RFU (those bits shall be 0)

Table 68: Verify Signature Init P1 Reference Parameter Codification.

Details of the Ubirch protocol are described at the following link:
https://developer.ubirch.com/utp.html#how-it-works-in-a-nutshell

Reference control parameter P2 shall indicate the Data Format and shall be coded according to the following table:

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
-	-	-	-	-	-	-	0	Entry ID + Algorithm
-	-	-	-	-	-	-	1	RFU
X	X	X	X	X	X	X	-	RFU (those bits shall be 0)

Table 69. Verify Signature Init P2 Reference Parameter Codification

The Verify Signature command data should be encoded according to the following table:

TAG	LEN	Value Description	Value	Presence
'C4'	Var.	Entry ID of Verifying Key/Certificate	'xxxx…'	Mandatory
'D0'	1	Algorithm to be used	See table: Verify Signature Init Supported Algorithms	Mandatory

Table 70. Verify Signature Init Command Data

Value	Meaning
'19'	ALG_HMAC_SHA_256
'1B'	ALG_HMAC_SHA_512
'21'	ALG_ECDSA_SHA_256

Table 71. Verify Signature Init Supported Algorithms

NOTES:

- Signature can be verified using either an existing Key or an existing Certificate inside Secure Storage.

- If Entry ID indicated is not either a Public or Symmetric Key or a certificate, an Error SW shall be returned.

Verify Signature Init Response data shall be empty if the command has been successfully executed.

Status conditions returned by the applet for Verify Signature Init:

SW	Reason
'90 00'	Command performed successfully.
'6A 86'	Incorrect parameter P1 or P2.
'6A 80'	Algorithm not supported.
	Entry ID and algorithm inconsistent
'6A 88'	Entry not found

Table 72. Verify Signature Init Status Conditions

2.2.4 Verify Signature Update/Final

Verify Signature Update/Final command is intended to verify a signature from the given payload using the information provided in a previous Verify Signature Init command.

The Verify Signature Update/Final command APDU format of the command is described as follows:

Command	CLA	INS	P1	P2	Lc	Data
Verify Signature Update/Final	'80'	'B8'	Verify Signature Update/Final P1	'00'	Var.	See table Verify Signature Update/Final Command Data

Table 73. Verify Signature Update/Final Command APDU format

Reference control parameter P1 shall be coded according to the following table:

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
1	-	-	-	-	-	-	-	Last (or only) block (Final Verify Signature)
0	-	-	-	-	-	-	-	More blocks (Update Verify Signature)
-	X	X	X	X	X	X	X	RFU (those bits shall be 0)

Table 74: Verify Signature Update/Final P1 Reference Parameter Codification

The Verify Signature Update/Final command data should be encoded according to the following table:

TAG	LEN	Value Description	Presence
'C5'	Var.	Plain data to be verified	Mandatory
'C6'	Var.	Signature	Mandatory

Table 75: Verify Signature Update/Final Command Data

NOTES:

- In case of using ALG_ECDSA_SHA_256 algorithm, the signature shall be encoded as an ASN.1 sequence of two INTEGER values, r and s.

Verify Signature Update/Final Response shall be encoded according to:

Name	Length	Value	Presence
Signature Verification Check	1	'00' if signature has not been successfully verified	Mandatory
		'FF' if signature has been successfully verified	

Table 76. Verify Signature Update/Final Response Data

Status conditions returned by the applet for Verify Signature Update/Final:

SW	Reason
'90 00'	Command performed successfully.
'6A 86'	Incorrect parameter P1 or P2.
'69 85'	Verify Signature Init not previously sent.
'6A 80'	Incorrect tag or tag already sent.
	Last command sent but there is data missing.

Table 77. Verify Signature Update/Final Status Conditions

About G+D Mobile Security

It's a long way to becoming a trusted identity service provider. Don't go it alone. G+D Mobile Security is part of the G+D Group with more than 11,000 employees worldwide. Our staff of 5,800 experts in over 50 sales and partner offices is glad to advise and support you with years of experience and comprehensive solutions that let you meet the challenges of a connected automotive industry and capitalize on its opportunities.

G+D Mobile Security

Giesecke+Devrient Mobile Security GmbH
Prinzregentenstrasse 159
81677 Munich
GERMANY
Phone: +49 89 41 19-0
www.gi-de.com/mobile-security
mobilesecurity@gi-de.com

About Ubirch

Ubirch is the specialist for blockchain-based technology in the field of IoT with locations in Cologne, Berlin and Munich. The team consists of experienced specialists in cryptography, blockchain and data-driven business models. Consisting of an extremely lightweight client for sensor firmware and the matching cloud backend, Ubirch's "Blockchain for Things" product enables ilitary-grade data protection to deliver new business models for the Internet of Things. Innovative cryptography and blockchain technologies guarantee the trustworthiness of IoT data. Ubirch is aimed primarily at customers in the industrial, smart cities, insurance, IoT start-ups, energy supply and logistics segments.

For further information visit www.ubirch.com