FINEID - S1 v2.1

Electronic ID Application

Application Note 1

Population Register Centre (VRK)
Certification Authority Services
P.O. Box 70
FIN-00581 Helsinki
Finland
http://www.fineid.fi

Authors

Name	Initials	Organization	E-mail
Antti Partanen	AP	VRK	antti.partanen@vrk.fi
Gemalto	GTO	Gemalto	

Document history

Version	Date	Editor	Changes	Status
1.0	5.12.2011	AP	Public edition	Final

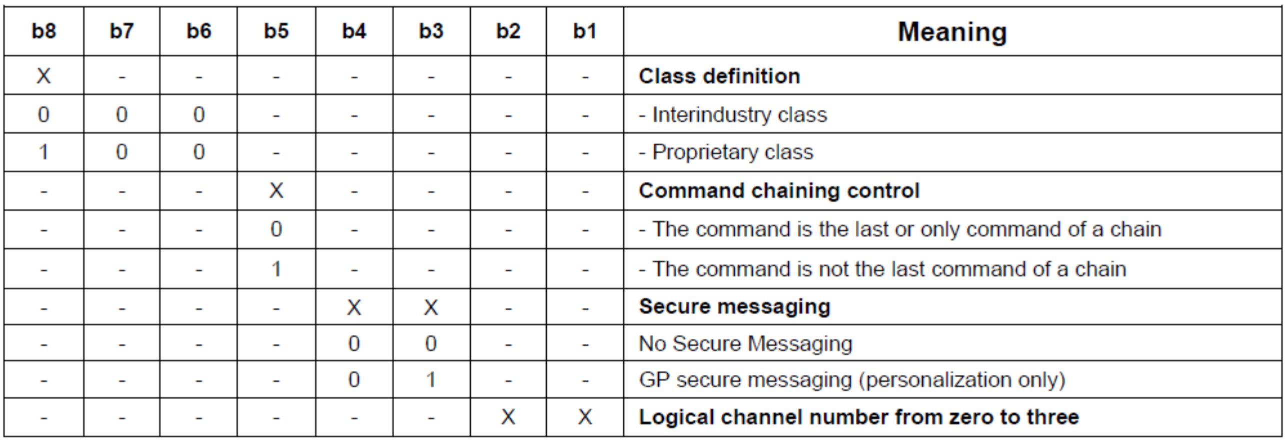
Table of Contents

EID2048 Applet: Application Note 1

1 Management of data when data size is larger than I/O buffer

1.1 CLASS byte coding

The class byte coding shall be as below:



b8	b7	b6	b5	b4	b3	b2	b1	Meaning
X	-	-	-	-	-	-	-	**Class definition**
0	0	0	-	-	-	-	-	- Interindustry class
1	0	0	-	-	-	-	-	- Proprietary class
-	-	-	X	-	-	-	-	**Command chaining control**
-	-	-	0	-	-	-	-	- The command is the last or only command of a chain
-	-	-	1	-	-	-	-	- The command is not the last command of a chain
-	-	-	-	X	X	-	-	**Secure messaging**
-	-	-	-	0	0	-	-	No Secure Messaging
-	-	-	-	0	1	-	-	GP secure messaging (personalization only)
-	-	-	-	-	-	X	X	**Logical channel number from zero to three**

If an error occurs during the check of the CLA byte of an APDU command, the applet must return the status RES_CLA_ERR.

1.2 Command chaining mechanism

If indicated for a command (b5 of the class byte), command chaining shall be supported according to ISO/IEC 7816 part 4. The commands that support command chaining are specified later in this application note.

If the data field length to send to the application is more than FFh bytes, the data field must be sent by blocks of data of FFh bytes or less. Each block is composed by the **same** command header (bit 5 of the class byte excepted) and by the block data field.

The CLA bit 5 of the last command of a chain shall be set to 0, whereas the bit 5 for all preceding chaining commands shall be set to 1.

The command headers are identical in all commands of a chain otherwise the card returns RES_INS_ERR.

If the application does not respond with RES_OK to a command of a chain with CLA '1X' (or ‘9X’), command chaining is aborted.

The command is processed only when all the data field has been received by the application (intermediate RES_OK only means correct command format)

If the command chaining mechanism is unavailable for a command, the card returns RES_CLA_ERR.

Example with a data field of 300d (12Ch) bytes:

Data field: M1 to M300

Command header: 00 INS P1 P2

The 2 sent commands could be: (the data field can be truncated “anywhere”)

10h INS P1 P2 F0h M1 .. M240 (The card must return RES_OK)

00h INS P1 P2 3Ch M241 .. M300 (The card processes the command with all the data field)

Retrieval of response data longer than 256 bytes

In the following use cases the application must return up to 256 bytes plain data plus TL structure so that the total length of the response data exceeds 256 bytes:

- PSO commands if a 2048-bit key is used.
- GET DATA command if a 2048-bit key is used.

This section defines how these cases shall be treated.

The way to retrieve all the outgoing data of such cases consists of a “Retrieval sequence”

- A Retrieval Sequence begins by a command sent to the card,
- A Retrieval Sequence finishes by the last Get Response command sent to retrieve the last outgoing data.

Byte transmission protocol T = 0

Case 2 command

When a case 2 command shall return more than 256 bytes, only the length Le = ‘00’ shall be authorized by the card, else the card returns RES_REDO_ERR.

For Le=0, the card returns 256 bytes and a status ‘RES_MORE + xx’ where ‘xx’ represents the remaining data (‘xx’ = ‘00’ if remaining data length ≥256 bytes). The retrieval sequence is open.

The command used to retrieve the remaining data is a Get Response command. For any subsequent Get Response command:

- If (Le > Licc with Licc < 256), the card returns RES_REDO_ERR + Licc. The retrieval sequence remains open.
- If Le = Licc and Licc < 256) or (Le=0 and Licc=256) the card returns Licc byte and a status RES_OK. The retrieval sequence is closed.
- If Le = 00 with Licc > 256, the card returns 256 bytes and the status ’ RES_MORE + xx’ where ‘xx’ represents the remaining data (‘xx’ = ‘00’ if remaining data length ≥ 256 bytes). The retrieval sequence is open.

Case 4 command

After the processing of the case 4 command has been completed without error, if outgoing data length > 256 bytes, the card returns RES_MORE.

For any subsequent Get Response command the process is the same as for the case 2.

2 Commands supporting command chaining mechanism

2.1 GET DATA

GET DATA command is used for retrieving a public key part of a RSA key pair. The file from which the key information is being retrieved must have been selected using the SELECT FILE command.

Note: If retrieving the public part of a 2048 RSA key pair, the chaining mechanism must be used.

Byte	Value
CLA	00h or 10h (if chaining mechanism used)
INS	CAh
P1	01h
P2	See Table 2
Lc	Empty
Data	Empty
Le	Number of bytes expected in response

Table 1. GET DATA command APDU

Coding of the P2									
b8	B7	b6	b5	B4	b3	b2	b1	Hex	Meaning
0	0	0	0	0	0	0	1	'00'	Key info: algorithm identifier, length of modulus and length of public exponent
0	0	0	0	0	0	1	0	'01'	Modulus
0	0	0	0	0	0	1	1	'02'	Public exponent
Any other value								-	RFU

Table 2. Coding of P2 of GET DATA command

GET DATA Response APDU	
Data field	See Table 4
SW1-SW2	Status bytes

Table 3. GET DATA response APDU

GET DATA Response APDU Data field			
Value of P1	Value of P2	Data field coding	
01h	00h	**Value**	**Length**
		algorithm identifier (‘92 00’ (RSA CRT) is the only currently supported value)	2
		bit length of modulus	2
		bit length of public exponent	2
01h	01h	**Value**	**Length**
		bit length of modulus	2
		Modulus	var
01h	02h	**Value**	**Length**
		bit length of public exponent	2
		Public exponent	var

Table 4. Response data field of GET DATA command

2.2 Applet version

To trace the functional applet version a dedicated TAG is added to the Get Data command.

In case of the couple P1-P2 parameter equals ‘DF’-‘30’, the applet version is returned on 5 bytes according to the following table:

Name	Value in ASCII	Length in bytes
V : 1 byte ASCII coded for version, RR : 2 bytes ASCII coded for release	‘v‘ | V | ’.’ | RR | ’	5

Table 5. Applet version

Example of version:
For an applet 1.25, the ASCII value is: ‘v1.25’ and the hexadecimal value is: ’76 31 2E 32 35’.

Success conditions for GET DATA		
‘61xx'	RES_MORE	xx data to available through Get Response command
‘9000'	RES_OK	OK

Table 6. GET DATA success conditions

Error conditions for GET DATA		
'6400'	RES_EXEC_ERR	Execution aborted (RSA file is deactivated)
'‘6982’	RES_AC_ERR	Security status not satisfied
‘6A81’	RES_FUNC_ERR	Function not supported (P1-P2 not recognised)
'6A88'	RES_NO_DATA_ERR	Referenced data not found (current file is DF or EF but not RSA key file, or RSA file is empty)
‘6Cxx’	RES_REDO_ERR	Wrong length (wrong L_e field; ‘xx’ indicates the exact length)

Table 7. GET DATA error conditions

2.3 PSO: CDS

PERFORM SECURITY OPERATION: COMPUTE DIGITAL SIGNATURE command computes a digital signature. The private key and algorithm to be used must be specified using the MANAGE SECURITY ENVIRONMENT command.

The input to the command may be either

- a hash code (e.g. SHA-1 hash value 20 bytes),
- a DigestInfo ASN.1 structure encapsulating the hash code, or
- a full modulus size input buffer (padding done by host application), or
- empty (hash code is calculated by preceding PSO: HASH command(s))

according to the selected algorithm reference value.

Byte	Value
CLA	00h or 10h (if chaining mechanism used)
INS	2Ah
P1	9Eh - digital signature data object is returned in response
P2	9Ah – data field contains data to be signed
Lc	Length of subsequent data field or empty
Data	If algorithm reference in SE = 00h - Data to be signed (e.g. encapsulated hash code). Padding is done to the full modulus length by the host application. If algorithm reference in SE = 02h: - Hash code encapsulated by the host application into DigestInfo structure. Padding is done internally by the card. If algorithm reference in SE = 12h - Hash code. Card encapsulates the hash into DigestInfo structure and pads it internally according to PKCS#1 v1.5 into full modulus length. Or - None. Hash is computed by preceding PSO: HASH command(s).
Le	Empty or maximum length of data expected in response

Table 8. PSO: COMPUTE DIGITAL SIGNATURE command APDU

Byte	Value
Data	Digital signature
SW1-SW2	Status bytes

Table 9. PSO: COMPUTE DIGITAL SIGNATURE response APDU

Success conditions for PSO: COMPUTE DIGITAL SIGNATURE		
‘61xx'	RES_MORE	xx data to available through Get Response command
‘9000'	RES_OK	OK

Table 10. PSO: COMPUTE DIGITAL SIGNATURE success conditions

Error conditions for PSO: COMPUTE DIGITAL SIGNATURE		
‘6700’	RES_LEN_ERR	Wrong length
'6982’	RES_AC_ERR	Security status not satisfied
'6984'	RES_REF_INVALID_ERR	Reference data invalidated (RSA file is deactivated)
'6985'	RES_COND_ERR	Conditions of use not satisfied (SE for operation not set correctly or hash not computed)
‘6A81’	RES_FUNC_ERR	Function not supported
‘6A86’	RES_PAR_ERR	Incorrect parameters P1-P2
‘6F00’	RES_GEN_ERR	No precise diagnosis is given

Table 11. PSO: COMPUTE DIGITAL SIGNATURE error conditions

2.4 PSO: DECIPHER

PERFORM SECURITY OPERATION: DECIPHER command decrypts an encrypted message (cryptogram). The key and algorithm to be used must be specified using the MANAGE SECURITY ENVIRONMENT command.

Note: If deciphering a message with 2048-bit keys, the chaining mechanism must be used.

Byte	Value
CLA	00h or 10h (if chaining mechanism used)
INS	2Ah
P1	80h – decrypted value is returned in response
P2	86h - data field contains padding indicator byte (00h according to ISO/IEC 7816-4) followed by the cryptogram (Note! If chaining mechanism is used padding indicator must be included only in the first command in chain)
Lc	Length of subsequent data field
Data	00h (padding indicator byte) || cryptogram (Note! If chaining mechanism is used padding indicator must be included only in the first command in chain)
Le	Empty or maximum length of data expected in response

Table 12. PSO: DECIPHER command APDU

Byte	Value
Data	Decrypted data
SW1-SW2	Status bytes

Table 13. PSO: DECIPHER response APDU

Success conditions for PSO: DECIPHER		
‘61xx'	RES_MORE	xx data to available through Get Response command
‘9000'	RES_OK	OK

Table 14. PSO: DECIPHER success conditions

Error conditions for PSO: DECIPHER		
‘6700’	RES_LEN_ERR	Wrong length
'6982’	RES_AC_ERR	Security status not satisfied
'6984'	RES_REF_INVALID_ERR	Reference data invalidated (RSA file is deactivated)
'6985'	RES_COND_ERR	Conditions of use not satisfied (SE for operation not set correctly or hash not computed)
‘6A81’	RES_FUNC_ERR	Function not supported
‘6A86’	RES_PAR_ERR	Incorrect parameters P1-P2
‘6A80’	RES_DATA_ERR	Incorrect parameters in data field (padding indicator or padding of deciphered data invalid)
‘6F00’	RES_GEN_ERR	No precise diagnosis is given

Table 15. PSO: DECIPHER error conditions

2.5 PSO: HASH

PERFORM SECURITY OPERATION: HASH command computes a hash sum. The algorithm to be used must be specified using the MANAGE SECURITY ENVIRONMENT command (using DST or HT CRDO in the data field). Currently only supported algorithm is SHA-1. This command supports command chaining mechanism, which utilizes the CLA value to indicate the end of the command chain. The command chain has CLA = 10h for all but the last command of the chain, which has CLA = 00h. In chained commands the commands with CLA = 10h shall carry only data quantities which are multiples of the block size of the hashing algorithm (64 bytes for SHA-1). The last command of the chain has no data length limitations. In order to be able to sign or verify the generated hash sum, the CLA must be 00h (end of chain) in the PSO: HASH command given immediately before the PSO: COMPUTE DIGITAL SIGNATURE command.

Byte	Value
CLA	00h or 10h (if chaining mechanism used)
INS	2Ah
P1	90h
P2	80h
Lc	Length of subsequent data field
Data	Data to be hashed
Le	Empty or maximum length of data expected in response.

Table 16. PSO: HASH command APDU

The data field may contain zero or more (plain value) bytes to be integrated into the hash sum (if no bytes are provided, the initial hash state is generated). Length of the data field shall be multiple of the block size of the hashing algorithm (64 bytes for SHA-1) for all but the last command of the chain.

For the further processing of the computed hash code the following cases have to be distinguished:

1. The hash code is stored in the card: the calculated hash code is stored in the card and available for use in a subsequent command (PSO: COMPUTE DIGITAL SIGNATURE). In this case the Le field of PSO:HASH command is empty and the algorithm identifier specified in the previous MSE:SET command (using the DST CRDO) shall be 12h. In this case it not possible to read out the generated hash sum.
2. The hash code is delivered by the card in the response. The Le field has to be set to the appropriate length (20 bytes for SHA-1).

Byte	Value
Data	Empty or calculated hash Empty: the used MSE:SET command before PSO:HASH contains DST CRDO Calculated hash: the used MSE:SET command before PSO:HASH contains HT CRDO
SW1-SW2	Status bytes

Table 17. PSO:HASH response APDU

Success conditions for PSO: HASH		
‘9000'	RES_OK	OK

Table 18. PSO: HASH success conditions

Error conditions for PSO: HASH		
‘6700’	RES_LEN_ERR	Wrong length
'6985'	RES_COND_ERR	Conditions of use not satisfied (SE for operation not set correctly or hash not computed)
‘6A81’	RES_FUNC_ERR	Function not supported
‘6A86’	RES_PAR_ERR	Incorrect parameters P1-P2
‘6A80’	RES_DATA_ERR	Incorrect parameters in data field (length of DO incorrect)

Table 19. PSO: HASH error conditions