A30

Secure Authenticator

Rev. 3.0 — 27 January 2025 Product data sheet
976730

1 General description

A30 is a secure authentication IC for IoT platforms, electronic accessories, and consumable devices such as home electronic devices, mobile accessories, and medical supplies.

A30 contains ECC key pairs, which can be generated by the IC itself to make sure that private keys are never exposed outside the IC. Also it performs cryptographic operations for security critical communication and control functions.

A30 has Common Criteria EAL 6+ security certification with AVA_VAN.5 on product level [1] and supports a generic Crypto API providing AES, ECDSA, ECDH, SHA, HMAC, and HKDF cryptographic functionality to users. Asymmetric cryptography features support 256-bit ECC over the NIST P-256 and brainpoolP256r1 curves. Symmetric cryptography features support both AES-128 and AES-256. Also it supports PKI-based mutual authentication including certificate handling. The CC security certification ensures that the IC security measures and protection mechanisms have been evaluated against sophisticated noninvasive and invasive attack scenarios.

A30 supports an I^2C contact interface with two GPIOs.

A30 supports a low-power design, and consumes only 5 µA at Halt mode when an external VDD is supplied.

2 Features and use cases

2.1 Use cases

A30 can be used for

- Secure key(s) and certificate(s) storage
- PKI (public key infrastructure) based authentication and communication
- Device only, device-to-device, device-to-cloud authentication
- Secure connection for consumer devices, industrial machines, and medical devices
- Battery passport and/or Digital product passport
- Device to meet strengthening cybersecurity requirements

2.2 Key features

A30 is designed to support many IoT applications and solves the problems in IoT applications' full life cycle.

- ECC key generation on the IC, and provisioning item level certificate(s) in NXP, or in the field.
- The following crypto primitives are supported: AES-128/256 (ECB, CBC, CMAC, CCM, GCM), ECDSA, and ECDH over NIST P-256 and brainpoolP256r1, SHA-256/384, HMAC, and HKDF.
  This allows to support advanced crypto protocols such as SIGMA-I, TLS1.3 and Matter.
- Nonreversible monotonic counter as the usage counter
- Delivery of the list of UID and certificates at shipping from NXP
- I^2C target operates at 100 kHz (standard mode), 400 kHz (fast mode), or 1 MHz (Fast-mode Plus)
- Two configurable GPIOs; 1 GPIO can be used for power downstream - up to 10 mW for batteryless applications
- 1 V operation with 1.5 V battery
- Small footprint on PCB with WLCSP16

2.3 Configuration

A30 can be used as an I^2C target with Host MCU.

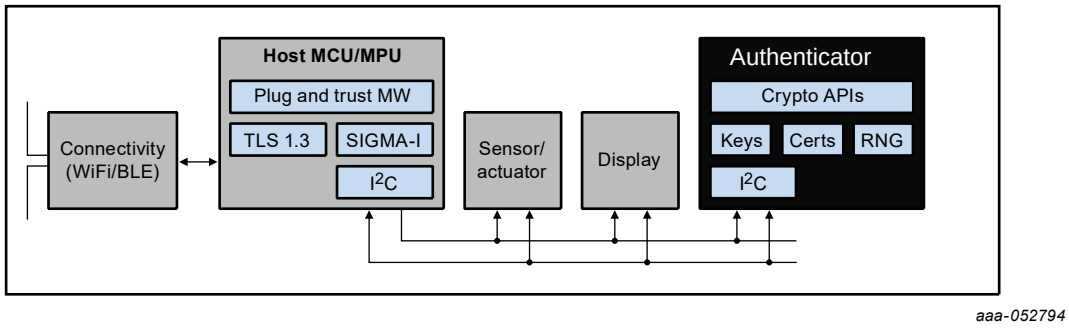


Figure 1. A30 solution block diagram

There are many configuration options to meet different types of applications.

2.4 Configuration as authenticator

A30 can be used for consumable authentication. MCU can read the certificate from A30 and do ECC-based authentication via ECDH, ECDSA, or full SIGMA-I protocol (Section 6.3.2).

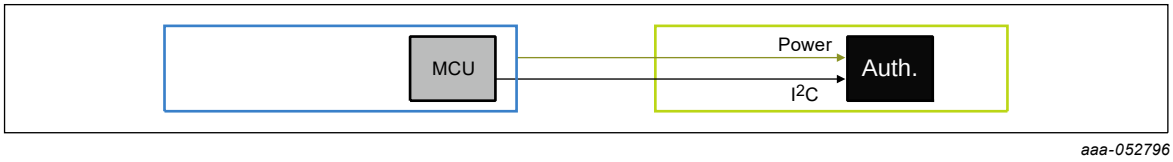


Figure 2. A30 for the consumable authentication

The user can check the originality of the consumable part, and get its status, for example, how many times the device has been powered up, or used with a nonreversible monotonic counter.

With this configuration, the target application is the accessory for mobiles or electronic devices. (For example, USB-C cable, Wireless charger.)

2.5 Configuration to secure IoT applications

A30 can be used for many other IoT applications.

With many other wired/wireless standards - WiFi, Bluetooth, ZigBee, Thread, A30 can be used to store keys and certificates securely, provide one-way and/or mutual authentication, and sign data being transferred.

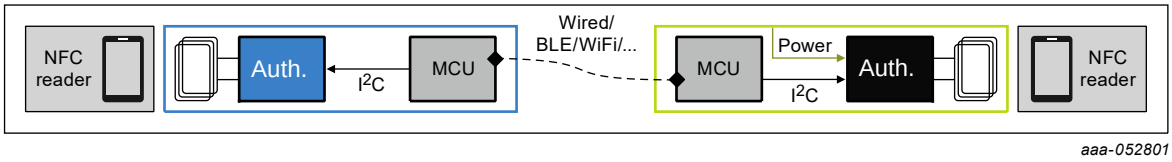


Figure 3. A30 solution block diagram

In this configuration, the target applications are IoT platforms supporting cloud onboarding and secure communications, for example, with Matter.

3 Ordering information

Table 1. Ordering information

Type number	Package		
	Name	Description	Version
A30LDJUK	WLCSP	A30, 16 KB memory	SOT2127-2
A30LDJHN2	HVQFN	A30, 16 KB memory	SOT917-6(DD)

4 Block diagram

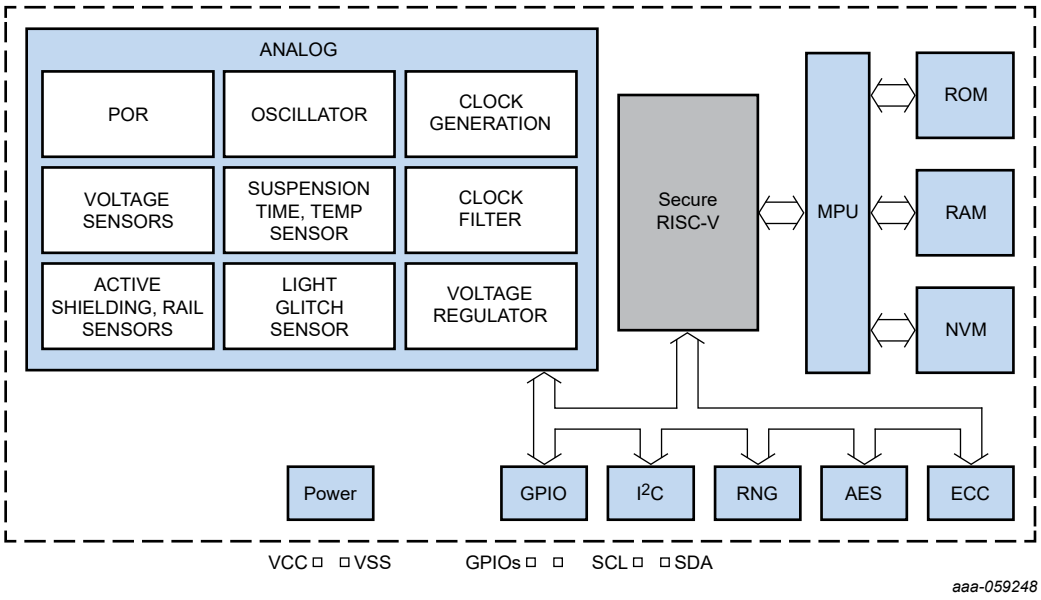


Figure 4. Block diagram

5 Pin description

A30 provides 6 pins:

Table 2. A30 pin configuration

Symbol	Description
V_{CC}	Logic and I^2C/GPIO power supply voltage input
V_{SS}	Ground
GPIO1	General Purpose IO
GPIO2	General Purpose IO
SDA	I^2C target data I/O
SCL	I^2C target clock input
RFU	To connected to Ground

6 Functional description

A30 supports I^2C interface with ISO/IEC 7816-4.

A30 adopts NTAG / NFC file system with a dedicated AID. NTAG, PCD, PICC, Reader, and all NFC related terms are used to represent NFC inherited authentication architecture.

6.1 I^2C support

A30 supports I^2C target communication with 7-bit target address according to [15].

The following bus speeds are supported, though potentially limited by pullup resistance and load capacitance depending on the HW configuration.

- 100 kHz (Standard-mode)
- 400 kHz (Fast-mode)
- 1 MHz (Fast-mode Plus)

At the data link layer, the T=1' protocol as specified in [16] is supported. Only the default parameter values are specified here.

6.1.1 I^2C parameter values

6.1.1.1 Target address

The default target address is 0x20.The target address can be changed through SetConfiguration Option 0x10, see Command SetConfiguration.

6.1.1.2 Communication interface parameters

The communication interface parameters (CIP) as defined by [16] are specified in Table 3.

Table 3. I^2C communication interface parameters

Name	Length	Description	Value
PVER	1	Protocol Version:[16] defines version '01' of the protocol.	0x01
Length of IIN	1	Length of Issuer Identification Number	0x04
IIN	3-4	Issuer Identification Number (according to [7812-1], BCD encoded)	0x63070093
PLD	1	Physical Layer ID: '01' for SPI / '02' for I^2C	0x02
Length of PLP	1	Length of Physical Layer Parameters	0x08
Configuration	1	Characteristics supported by SE: b1= 0: Clock stretching not supported Other bits: RFU	0x00
PWT	1	Power wake-up Time (ms)	0x02
MCF	2	Maximal Clock Frequency at which the SE may operate (in kHz)	0x03E8 (1 MHz)
PST	1	Power-Saving Time-outs (in ms)	0x00
MPOT	1	Minimum Polling Time (conditional to Polling Mode support) (in ms)	0x01
RWGT	2	R/W Guard Time (in µs)	0x0064
Length of DLLP	1	Length of Data Link Layer Parameters	0x04
BWT	2	Block Waiting Time (in ms)	0x03E8 (ca. 1 sec)

Table 3. I²C communication interface parameters...*continued*

Name	Length	Description	Value
IFSC	2	Maximum Information Field Size of the SE (in bytes) (i.e. initial value)	0x00FE
Length of HB	1	Length of Historical Bytes (max. 32 bytes)	0x00
Historical Bytes	Var	Empty	-

PWT value does not depend on whether the Halt watchdog Timer (HWDT) has been enabled with SetConfiguration Option 0x14, see Command SetConfiguration.

6.1.2 I²C Application Remarks

6.1.2.1 Power Management

A30 contains an adaptive power management system reducing or stopping internal clocks.

In case the internal clock is stopped A30 might not be able to serve the I²C bus while the internal clock is stopped. In this case the host will read 'FF' while the internal clock is stopped.

This cases will be detected with high probability by the CRC check.

The recommended error-recovery on failed CRC checks is as following:

1. Read IFSC number of bytes to clear before continuing.

2. Send R-Block CRC Fail to Card

In case the length information bytes are read as FF the host protocol stack shall abort reading after the maximum frame size (254+6 bytes) supported by A30, e.g. by checking if the response is longer than IFS + protocol overhead.

6.1.2.2 Write after Write behavior

For Write after Write with two correct transmit messages the device response is discarded when the new message is received. Instead of the expected read message an error message A5-82-00-00-89-E0 (Other Error).

6.2 Command format and chaining

6.2.1 Native command format

A30 always communicates in ISO/IEC 7816-4 wrapped mode as described in Section 6.2.2. Nevertheless, it is important to understand the basic format of native commands which consist of the following parts.

A command as sent by the PCD consists of the concatenation of:

- the command code (Cmd)
- zero, one or more header fields (CmdHeader)
- zero, one or more data fields (CmdData)

The response as sent by the PICC consists of the concatenation of:

- the return code (RC)
- zero, one or more data fields (RespData)

A30 supports the APDU message structure according to ISO/IEC 7816-4 [4] for:

- wrapping of the native command format into a proprietary ISO/IEC 7816-4 APDU
- a subset of the standard ISO/IEC 7816-4 commands (ISOSelectFile, ISOReadBinary, ISOUpdateBinary)

Remark: Communication via native ISO/IEC7816-4 commands without wrapping is not supported.

On the native command interface, plain command parameters consisting of multiple bytes are represented least significant byte (LSB) first. Similar as for ISO/IEC 14443 parameters during the activation, see [2]. For cryptographical parameters and keys (including the random numbers exchanged during authentication, the TI and the computed MACs), this does not hold. For these, the representation on the interface maps one-to-one to the most significant byte (MSB) first notation used in this specification.

Within this document, the '0x' prefix indicates hexadecimal integer notation, i.e. not reflecting the byte order representation on the command interface at all.

6.2.2 ISO/IEC7816-4 communication frame

A30 uses ISO/IEC 7816-4 [4] type APDUs for command-response pair for both, wrapping of native commands, as outlined in Section 6.2.1 and standard ISO/IEC 7816-4 commands.

For all parameters of standard ISO/IEC 7816-4 commands, the representation on the interface is most significant byte (MSB) first notation. As data like the 2-byte ISO/IEC 7816-4 file identifiers, are in different order for the wrapped native commands, this needs to be taken into account.

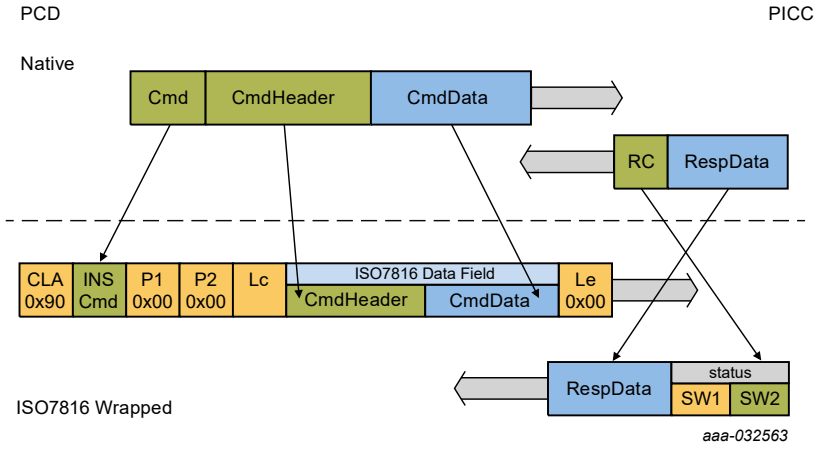


Figure 5. ISO/IEC 7816-4 command response pair

Table 4. ISO/IEC 7816-4 command fields

Field	Description	Length
Command header	Class byte (CLA)	1
	Instruction (INS)	1
	Parameters (P1,P2)	2
Lc field	Length of command data field (Lc), absent if no data field is present	1, 3
Command data field	Absent if no data is sent in the command	Lc
Le field	Expected response length. If Le is 0x00, then all available data is sent back for ISO/IEC 7816-4 standard commands. For wrapped commands, Le must always be set to 0x00.	1, 2, 3

In general, A30, supports Extended Length fields for Lc and Le, see [4]. However, for some commands the supported input size is restricted as specified in the command definition.

Table 5. ISO/IEC 7816-4 response fields

Field	Description	Length
Response data field	Response data if any, absent if no response data	up to Le
Response trailer	status byte (SW1SW2)	2

The field length and presence might vary for different commands, refer to the specific command description in Section 7.

6.2.3 Command chaining

A30 supports standard ISO/IEC 14443-4 [3] command chaining in the following cases:

- the PICC supports ISO/IEC 14443-4 chaining to allow larger command or response frames than the supported buffer size for variants of the following commands:
  - Native commands wrapped into ISO/IEC 7816-4 APDU: ReadData, WriteData, see Section 7.
  - Standard ISO/IEC 7816-4 commands: ISOReadBinary, ISOUpdateBinary
    i.e. every command where a larger frame size can occur.
- the PICC automatically split a response in several frames to fit with the FSD frame size supported by the PCD and communicated in the RATS.

When a PCD applies ISO/IEC 14443-4 chaining, see [3], it must assure the reassembled INF field containing the command header (i.e. ISO/IEC 7816-4 header bytes and/or (Cmd || CmdHeader)) fits within the PICC's buffer (FSC) communicated in the ATS. If not, the PICC may respond with LENGTH_ERROR.

The ISO/IEC 14443-4 chaining does not influence the secure messaging. This means that the secure messaging mechanisms are applied as if the command or response would have been sent in a single large frame. With regard to command execution, commands are handled as if they were received in one large frame, except for write commands where the total frame size can be larger than the supported FSC (WriteData and ISOUpdateBinary). In this case, command execution is started before the complete command is received.

For single frame write operations or chained frames that fit within the supported FSC it is ensured that either the data is completely written or not at all.

6.3 Authentication and Secure Messaging

6.3.1 Authentication overview

The A30 supports several mutual authentication protocols:

- symmetric mutual authentication: this authentication is initiated by AuthenticateEV2First or AuthenticateEV2NonFirst. The protocol is inherited and compatible with NTAG42x and MIFARE DESFire. It is based on AES-128 or AES-256.
- asymmetric mutual authentication: this authentication is initiated by ISOGeneralAuthenticate. It is based on 256-bit ECC.

Both mutual authentication methods initiate an EV2 secure messaging channel, see Section 6.3.6 based on AES-128 or AES-256 session keys.

Each authentication option can be used with different keys and on different I/O interfaces. However, the A30 only supports a single authentication session. The authentication session applies to the I/O interface, which

opened it. The other I/O interface has no current authentication session. The current session shall be closed if any of the following occur:

- a new mutual authentication is initiated (on either interface)
- the NTAG application is selected (on either interface)
- the key used to open the session is changed (for symmetric mutual authentication)
- the device enters HALT state
- the device is reset
- the OS processes an erroneous command on the interface, which opened the authentication session

The fundamental states as listed below are introduced in Figure 6.

- VCState.NotAuthenticated: This is the default state where there is no active authentication. The AuthKey is invalidated in this state. This state is reached after POR and activation.
- VCState.PartiallyAuthenticated: In this state, an authentication is ongoing. The A30 is expecting the second part. This means that any previous active authentication has already been lost.
- VCState.AuthenticatedAES: there is an active authentication reached by successfully executing the symmetric authentication protocol initiated with AuthenticateEV2First or AuthenticateEV2NonFirst. EV2 Secure Messaging, as defined in Section 6.3.6, is active. The targeted key of the last authentication is remembered as an AuthKey. Depending on these key access rights to subsequent commands may be granted or not.
- VCState.AuthenticatedECC: there is an active authentication reached by successfully executing the asymmetric mutual authentication protocol initiated with the ISOGeneralAuthenticate (CLA 0x00, INS 0x86, targeting a Sigma-I protocol). Also here, symmetric AES-based EV2 Secure Messaging, as defined in Section 6.3.6, is active. Access rights in this state depend on the targeted CARootKey and/or reader certificates presented during the authentication, see Section 6.4.2 and Section 6.4.3.

The transitions to and from those states are related to the secure messaging specification.

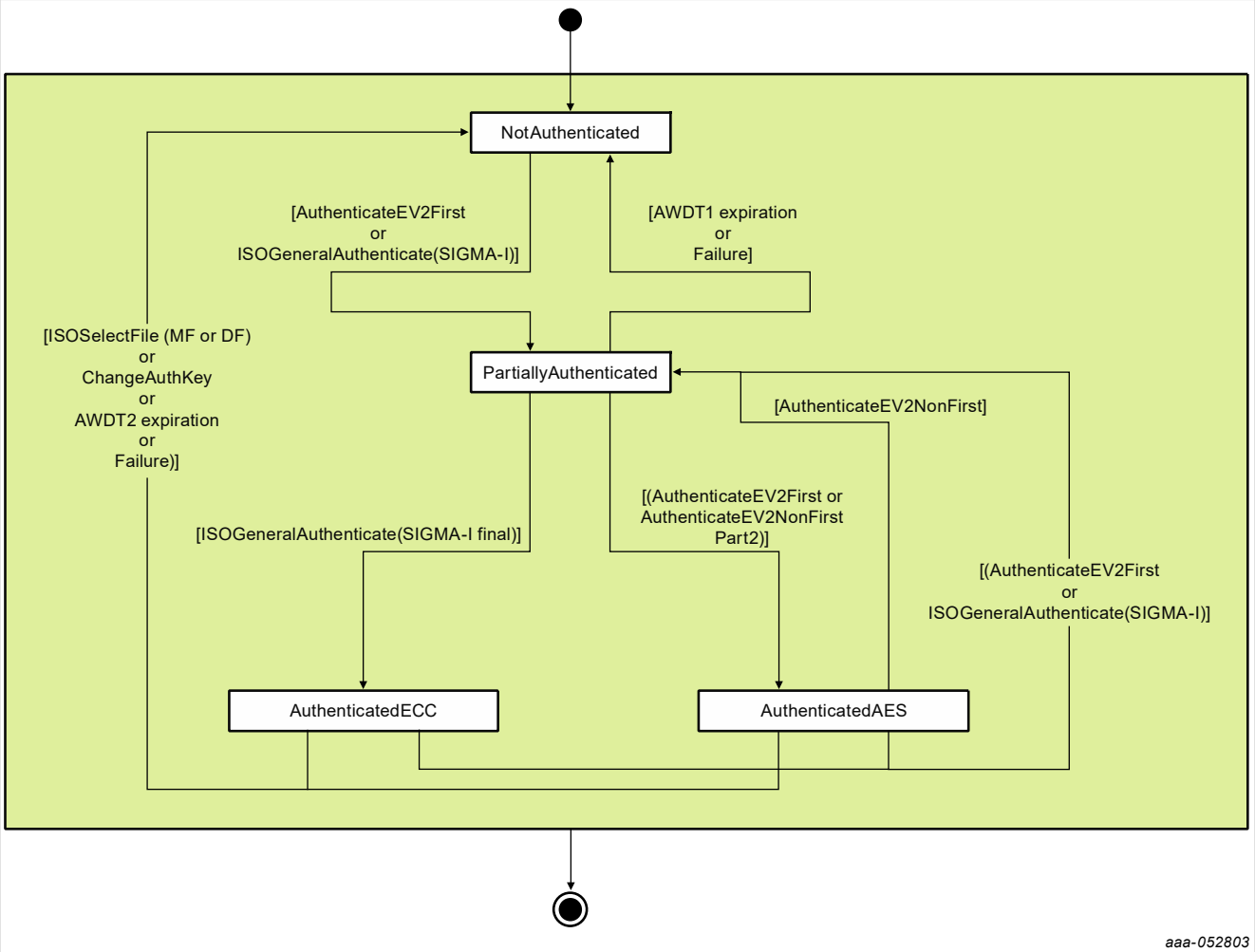


Figure 6. Authentication State Diagram

Here are the notes for Figure 6:

- Failure indicates any error: the A30 switches to the VCState.NotAuthenticated. In these cases the response is already sent with CommMode.Plain (as always in VCState.NotAuthenticated).
- If enabled, AWDT2 expiration aborts an ongoing authentication attempt, moving the A30 back from VCState.PartiallyAuthenticated to VCState.NotAuthenticated.
- If enabled, AWDT1 expiration aborts an active authentication session, moving the A30 back from VCState.AuthenticatedAES or VCState.AuthenticatedECC to VCState.NotAuthenticated.
- The authentication process consists of two parts. If only the first part is received, then any command different from the expected second part results in a failure.
- In both, VCState.AuthenticatedAES and VCState.AuthenticatedECC, the same AES-based secure messaging applies.
- ChangeAuthKey indicates ChangeKey targeting the currently authenticated key.

6.3.2 SIGMA-I authentication with ISOGeneralAuthenticate

The A30 supports an asymmetric based authentication protocol. Asymmetric protocol exchanges are made via the ISOGeneralAuthenticate command outlined in Section 7.3.1. Sending the ISOGeneralAuthenticate command to initiate a SIGMA-I mutual authentication resets any ongoing mutual authentication exchange or already established secure channel session. The NTAG application must be selected before asymmetric protocol execution can commence.

The Sigma-I protocol consists of an exchange of three messages between the *Initiator (or SIGMA-I Verifier)* and the *Responder (or SIGMA-I Prover).* In addition, if the certificates required are not found in the certificate cache (or if caching is not supported) then certificate request and reply messages are exchanged.

The SIGMA-I protocol can be executed with the host as the initiator (or SIGMA-I Verifier) or the A30 as the initiator (with host as SIGMA-I Prover). The data format shall remain consistent no matter which role the A30 plays.

If SIGMA-I Prover is used as the session protocol, then the host acts as the protocol responder. However, in this case, the host still needs to send the first command, initiating the message exchange.

The access rights granted to the host are by default the rights associated with the CA root public key used to validate the host's certificate chain. However, these rights can be reduced by the certificate issuer via a proprietary x.509 certificate extension. A certificate shall never have more access rights than its parent certificate.

6.3.2.1 Session keys

As part of the protocol, both sides generate shared session keys and IV (nonce value) as follows (see Section 6.3.2.5 for session key generation details):

Table 6. SIGMA-I Session Keys

Item	Description
K_e1	Encryption/Decryption key for message exchange
K_m1	MAC key used to generate input for session signature
IV_e1	AES CCM NONCE incremented for each message

6.3.2.2 Message types

Each SIGMA -I message has a TLV structure, where the tag indicates the type of message, and the value component is the payload. The message payload may be in plaintext, encrypted using AES-CCM, or a mixture of both, depending on the TLV tag.

The following table lists the message types, corresponding tags, and session keys to be used for encryption/decryption.

Table 7. SIGMA-I Message Types

Message	TLV Tag	Description	Payload
MSGI_PUBLIC_KEY	A0	Initiator sends its supported AES key sizes and its ephemeral public key	Protocol Options byte and Ephemeral ECDH public key (xP) in plaintext
MSGI_HASH_AND_SIG	A1	Initiator sends certificate hash (or full certificate) and signature	Cert hash (or full certificate) and signature encrypted with <K_e1, IV_e1>
MSGI_CERT_REQUEST	A2	Initiator requests a certificate from responder	Cert request message encrypted with <K_e1, IV_e1>

Table 7. SIGMA-I Message Types*...continued*

Message	TLV Tag	Description	Payload
MSGI_CERT_REPLY	A3	Initiator sends its certificate to responder	Certificate (optionally compressed), encrypted with <K_e1, IV_e1>
MSGI_ABORT_SESSION	AF	Initiator aborts protocol	None
MSGR_START_ PROTOCOL	B0	Host as responder hands control to device/initiator, to start protocol	None
MSGR_HASH_AND_SIG	B1	Responder sends the session AES key size and its ephemeral public key, certificate hash and signature	Session symmetric key size and Public key (yP) in plaintext, cert hash[1] and signature encrypted with <K_e1, IV_e1>
MSGR_CERT_REQUEST	B2	Responder requests a certificate from initiator	Certificate request message encrypted with <K_e1, IV_e1>
MSGR_CERT_REPLY	B3	Responder sends its certificate to initiator	Certificate (optionally compressed) encrypted with <K_e1, IV_e1>
MSGR_ABORT_SESSION	BF	Responder aborts protocol	None
MSG_SESSION_OK	B4	Device is responder, returns control to host upon successful authentication. Secure tunnel rules now apply.	None

[1] cert hash is a hash over the complete certificate including the Signature field.

An abort message shall be sent by the card in the following scenarios:

- Abort message is received from the host.

- Host certificate chain is syntactically correct but CA root public key can't be located to verify it.

- Session key size can't be mutually agreed.

The payload of each message used during protocol exchange may be wrapped with tags that identify the contents, as shown in the following table:

Table 8. Asymmetric authentication Protocols Payload Encodings

Tag	Length	Description
0x80	0x00	Certificate request (leaf, level = 0)
0x81	0x00	Certificate request (parent, level = 1)
0x82	0x00	Certificate request (parent, level = 2)
0x83	0x01	AES key size options
0x84	0x20	Certificate Hash
0x85	0x40	ECC Signature
0x86	0x41	Ephemeral ECDH public key, plaintext, uncompressed format
0x87	<var>	Encrypted payload
0x7F21	<var>	Uncompressed certificate

6.3.2.3 Protocol exchange – Host as initiator

When the host is the initiator (SIGMA-I Verifier) and the device is the responder, the messages fall evenly into APDU command and response:

Table 9. A30 as SIGMA-I responder

Message	Contents
Public key ➔ (C-APDU) data field	A0 46 83 01 <key sizes supported> (see Table 11). 86 41 04 <xP, public key, 64 bytes>
Cert hash and signature ➔ (R-APDU) data field	B1 81 B0 83 01 <key size selected> (see Table 11). 86 41 04 <yP, public key, 64 bytes> 87 68 <C_k_r (see Section 6.3.2.6): encrypted hash and signature>

Initiator
Cert request (optional)
➔ (C-APDU)
	A2 0C 87 0A
	// leaf cert request: 80 00 or
	// p1 cert request: 81 00 or
	// p2 cert request: 82 00
	<encrypted cert request>
	AES_CCM_Dec(K=K_e1, N=++IV_e1, A=NULL, C=Encrypted Cert Request)

Responder Cert reply (optional) ➔ (R-APDU)	B3 82 xx xx // uncompressed cert: 7F 21 <cert> <encrypted certificate> AES_CCM_Enc(K=K_e1, N=++IV_e1, A=NULL, P=Certificate Info)
Cert hash and signature ➔ (C-APDU)	A1 68 <C_k_i: encrypted hash and signature> AES_CCM_Dec(K=K_e1, N=++IV_e1, A=NULL, C=C_k_i)

Responder
Cert request (optional)
➔ (R-APDU)
	B2 0C 87 0A
	// leaf cert request: 80 00 or
	// p1 cert request: 81 00 or
	// p2 cert request: 82 00
	<encrypted cert request>
	AES_CCM_Enc(K=K_e1, N=++IV_e1, A=NULL, P=Certificate Request)

Initiator Cert reply (optional) ➔ (C-APDU)	A3 82 xx xx // uncompressed cert: 7F 21 <cert> <encrypted certificate> AES_CCM_Dec(K=K_e1, N=++IV_e1, A=NULL , C=Encrypted Certificate)

Table 9. A30 as SIGMA-I responder*...continued*

Message	Contents

End Session
🡐 (R-APDU)
	B4 00
	// **mutual authentication is complete**
	// session keys k_e2, k_m2 can be used
	// to send messages in secure tunnel

6.3.2.4 Protocol exchange – Host as responder

When the host is the responder (SIGMA-I Prover), it first transfers control to the device (initiator) by sending a control transfer message in C-APDU. The message contents are identical to the previous section, but the placement in C-APDU vs. R-APDU is reversed.

Table 10. A30 as SIGMA-I initiator

Message	Contents
Transfer control 🡐 (C-APDU) data field	B0 00
Public key 🡒 (R-APDU) data field	A0 46 83 01 <key sizes supported> (see Table 11) 86 41 04 <xP, public key, 64 bytes>
Cert hash and signature 🡐 (C-APDU)	B1 81 B0 83 01 <key size selected> (see Table 11) 86 41 04 <yP, public key, 64 bytes> 87 68 <C_k_r: encrypted hash and signature> AES_CCM_Dec(K=K_e1, N=IV_e1, A=NULL, C=C_k_r)

Initiator
Cert request (optional)
🡒 (R-APDU)
	A2 0C 87 0A
	// leaf cert request: 80 00 or
	// p1 cert request: 81 00 or
	// p2 cert request: 82 00
	<encrypted cert request>
	AES_CCM_Enc(K=K_e1, N=++IV_e1, A=NULL, P=Cert Request)

Responder Cert reply (optional) 🡐 (C-APDU)	B3 82 xx xx // uncompressed cert: 7F 21 <cert> <encrypted certificate> AES_CCM_Dec(K=K_e1, N=++IV_e1, A=NULL, C=Encrypted Certificate)
Cert hash and signature 🡒 (R-APDU)	A1 68 < C_k_i (see Section 6.3.2.6) encrypted cert hash and signature>

Table 10. A30 as SIGMA-I initiator*...continued*

Message	Contents

Responder
Cert request (optional)
🡐 (C-APDU)
	`B2 0C 87 0A`
	`// leaf cert request: 80 00 or`
	`// p1 cert request: 81 00 or`
	`// p2 cert request: 82 00`
	`<encrypted cert request>`
	`AES_CCM_Dec(K=K_e1, N=++IV_e1, A=NULL, C=Encrypted Certificate Request)`

Initiator
Cert reply (optional)
🡒 (R-APDU)
	`A3 82 xx xx`
	`// uncompressed cert: 7F 21 <cert>`
	`// compressed cert: 7F 22 <comp-cert>`
	`<encrypted certificate>`
	`AES_CCM_Enc_(K=K_e1, N=++IV_e1, A=NULL, P=Certificate Info)`

	`// mutual authentication is complete`
	`// this is known implicitly by both sides`
	`// session keys k_e2, k_m2 can be used`
	`// to send messages in secure tunnel`

6.3.2.5 SIGMA-I session key generation

Session key generation requires the A30's ephemeral private key and the host's ephemeral public key. The ECC domain curve to use for session key generation shall match the domain curve used to sign the A30's session signature. This is defined by the targeted certificate repository.

The session key generation process is as follows:

- Validate the host's public key
- Compute shared secret using ECDH with the private key of the A30 and the public key of the Host.
- Select AES session key size (AES-128 or AES-256). This shall be the largest key size mutually supported by both initiator and responder. If no mutually supported key size then the mutual authentication session is aborted with a protocol error. Key size definitions are outlined in Table 11.
- Generate session keys and IV used for mutual authentication
- Generate session keys used for the secure tunnel

Table 11. SIGMA-I Session Key Sizes

b7	b6	b5	b4	b3	b2	b1	b0	Description
-	-	-	-	-	-	-	x	AES-128
-	-	-	-	-	-	x	-	AES-256
x	x	x	x	x	x	--		RFU

The KDF algorithm is NIST SP800-108 compliant [9] and uses Counter Mode with AES as the PRF. The key size to use for the PRF shall match the session key size selected. As a PRF AES CMAC is used. SP 800-108 states counter-based KDF as follows:

K(i) := PRF (Ki, [i]2 || Label || 0x00 || Context || [L]2)

Ki: the base key shall be used for generation of all session keys and IVs. This key shall be derived using SHA-256(trans_xy) where trans_xy = *x-coordinate of shared secret* | *initiator's public key* | *responder's public*

key. When AES-256 is selected, then the complete 32 bytes shall be used; for AES-128 the derived bytes shall be truncated to the first 16 bytes.

i: two-byte iteration counter starting at 1. When AES128 is selected only 0x0001 is used; for AES 256 0x0001 and 0x0002 counter values are used and the output is concatenated.

Label: each mutual authentication key to be generated shall have a unique label:

- "K_e1", see Section 6.3.2.1
- "K_m1", see Section 6.3.2.1
- "IV_e1", see Section 6.3.2.1
- "K_e2" for EV2 ENC, see Section 6.3.6
- "K_m2" for EV2 MAC, see Section 6.3.6

Context: "SIGMA-I" for mutual authentication keys, "IVs" for mutual authentication IV or "EV2" for EV2 tunnel session keys

L: an integer specifying the output data length:

- 0x0100 AES-256 key
- 0x0080 for an AES-128 key
- 0x0068 for an IV_e1.
  Note: *The CCM nonce N is composed of the 13 leftmost IV_e1 bytes.*

6.3.2.6 A30 Signature generation

The A30 generates a unique signature every session, which is sent in an encrypted payload and also includes the leaf certificate hash of thwe A30. Signature generation and subsequent encryption depend on the role assumed by the A30. The private key and associated repository to use are either explicitly stated or the certificate repository with the lowest Id, which supports SIGMA-I shall be used.

The signature generation and encryption methodology are as follows. ECDSA-Sign and ECDSA-Verify is ECDSA Digital Signature Generation and Verification as defined in [23]. The hash function to be applied is SHA-256, as specified in NIST FIPS 180-4 [17]. AES-CMAC is according to [7] and AES_CCM is according to [25]. The AES CCM parameters according to the formatting Appendix A from [25] are a 2-byte length field *q*, a 13-byte nonce *n*, and an 8-byte tag *t*.

The following parameters are used:

- sk_init and sk_resp are the targeted private keys.
- keySize_init and keySize_resp are the initiator key sizes supported byte and responder key size selected byte, according to Table 11
- xP and yP are respectively the host/initiator's ephemeral public key and A30/responder's ephemeral public key
- leaf_cert_hash is SHA-256 of the end-leaf certificate of the A30 including the signature.
- A (Associated Data from [25]) is optional additional authenticated data (which is not encrypted) and is not applicable for this SIGMA-I implementation.

6.3.2.6.1 A30 as initiator

- *Init_ECC_Sig = ECDSA-Sign(sk_init , 0x02 || keySize_init || keySize_resp || yP || xP || AES_CMAC(K_m1, 0x02'|| leaf_cert_hash))*
- *Data = leaf_cert_hash || Init_ECC_Sig*
- *C_k_i = AES_CCM_Enc(K=K_e1, N=IV_e1, A=NULL, P=Data)*

6.3.2.6.2 A30 as responder

- *Resp_ECC_Sig = ECDSA-Sign(sk_resp, 0x01 || keySize_init || keySize_resp || xP || yP || AES-CMAC(K_m1, 0x01 || leaf_cert _hash))*
- *Data = leaf_cert_hash || Resp_ECC_Sig*
- *C_k_r = AES_CCM_Enc(K=K_e1, N=iv_e1, A=NULL, P=Data)*

6.3.2.7 SIGMA-I: Verification of the host

The A30 receives the end-leaf certificate hash and a session ECC signature from the Host. To authenticate the Host, the A30 shall verify the session signature using a trusted public key. The trusted public key can either be a prevalidated key stored in the A30's certificate cache or a key authenticated through validation of the associated public key certificate. If certificate caching is disabled or the public key isn't present in the A30's cache then the A30 shall request the host to provide public key certificates until the certificate chain of the leaf public key can be verified. The maximum depth of a certificate chain is 4, therefore, the A30 shall request up to a maximum of three certificates from the host (leaf, P1 and P2). If the leaf certificate public key of the Host cannot be validated then the authentication session is terminated.

Once the public key of the Host is validated, the A30 verifies the signature from the Host as follows. ECDSA-Verify is ECDSA Digital Signature Generation as defined in [23]. The hash function to be applied is SHA-256, as specified in NIST FIPS 180-4 [17]. AES-CMAC is according to [7].

The following parameters are used:

- pk_init and pk_resp are the targeted public keys, retrieved from the certificate chain.
- sig_init and sig_resp are the received signatures
- keySize_init and keySize_resp are the initiator key sizes supported byte and responder key size selected byte, according to Table 11
- xP and yP are respectively the A30/initiator's ephemeral public key and host/responder's ephemeral public key
- leaf_cert_hash is SHA-256 of the host's end-leaf certificate including the signature.

When the host's session signature is validated, the host is granted the access rights (from '0' to 'D') associated with the CA root public key used to validate the host's certificate chain (or restricted subset as specified in x.509 certificate extension).

6.3.2.7.1 A30 as initiator

- *ECDSA-Verify(pk_resp, sig_resp, 0x01 || keySize_init || keySize_resp || xP || yP || AES-CMAC(K_m1, 0x01 || leaf_cert_hash))*

6.3.2.7.2 A30 as responder

- *ECC_Verify(pk_init, sig_init, 0x02 || keySize_init || keySize_resp || yP || xP (|| AES-CMAC(K_m1, 0x02 || (leaf_cert_hash)))*

6.3.3 ECC-based card-unilateral authentication

A30 supports an ECC-based card-unilateral authentication protocol as described in this section. This allows for authenticating the card without requiring an authentication from the reader side. This protocol can be applied for Originality Check purposes, i.e. to ensure the genuineness of A30 ICs, as described in Section 6.16.1. This protocol does not open a secure messaging session.

The protocol can be executed with ISOInternalAuthenticate.

As the protocol creates a trace that cannot be repudiated, the privacy implications of enabling the feature should be evaluated.

6.3.3.1 Data structures and notations

6.3.3.1.1 ECCKey pair

The card-unilateral authentication applies a static key pair (*Priv.B, Pub.B*) from which the private key *Priv.B* is stored on the card and used by the card during the protocol.

6.3.3.1.2 Certificate

For the protocol, the public key *Pub.B* to be used by the reader for validating the authenticity of the card, should be authenticated through a certificate or certificate chain. This certificate (chain) can be stored on the card in a FileType.StandardData file and retrieved via the related commands before executing the ISOInternalAuthenticate.

For Originality Check purposes, the certificate is trust-provisioned during manufacturing, as described in Section 6.16.1.

During the further description of the protocol, the certificate validation is kept out of scope.

6.3.3.2 Cryptographic primitives

6.3.3.2.1 Elliptic Curve Digital Signature Generation and Verification

The card-unilateral authentication is based on the ECDSA Digital Signature Generation and Verification as defined in [12]. The hash function to be applied is SHA-256, as specified in NIST FIPS 180-4[17].

The following notations are used:

$Sig.B = ECDSA_{Sign}(Priv.B, M)$

$[true, false] = ECDSA_{Verify}(Pub.B, M, Sig.B)$

In the above example, *B* signs the message *M* with his private key *Priv.B*, resulting in the signature *Sig.B*. *Sig.B* consists of two integers (*Sig.B.r, Sig.B.s*) of a size equalling the curve size, i.e.both 32 bytes for an ECC-256 curve, resulting in a 64-byte signature. With $ECDSA_{Verify}$, the *Sig.B* is verified to be correct for the message *M* with the public key *Pub.B*, resulting in *true* or *false*.

6.3.3.3 ISOInternalAuthenticate

The authentication is initiated by ISOInternalAuthenticate. A detailed command definition can be found in Table 48.

The protocol can only be executed in VCState.NotAuthenticated and does not change the authentication state. All parameters in the command and response data field are BER-TLV data objects (DOs) encoded according to ISO/IEC7816-4 [3] with DER length encoding. Authentication DOs are collected under the 0x7C tag according to ISO/IEC 7816-4, Table 100. Other parameters use a context-specific tag according to ISO/IEC 8825-1 [18]. All DOs must be sent in the order specified in the command tables.

Upon reception of ISOInternalAuthenticate, the PICC checks the `ECCPrivateKey` addressed by P2 if the key does not exist or is not enabled for ECC-based unilateral authentication, the command is rejected. If the targeted `ECCPrivateKey` has an enabled `KeyUsageCtrLimit` that was already reached, see Section 6.7.1.2, the command is also rejected.

A30 supports ISOInternalAuthenticate at the PICC level by default for originality checking with the Section 6.16.1.1 at KeyNo 0x01. The command can be disabled for privacy purposes through SetConfiguration Option 0x0E. At the application level, it depends on the key policy configured during key creation or update, whether the protocol is supported for a specific key.

Certificates related to the unilateral authentication can either be stored in a certificate repository or in FileType.StandardData files.

The parameter `OptsA` is optional. As it may be used for potential future extensions, the current implementation accepts and ignores it, including TLV-structures with a bigger length. If present, *`OptsA`* is included in the signature calculation to allow protection against future protocol downgrade attacks. Future implementations may then also return an *`OptsB`* with Tag 0x80.

Upon reception of the command, the PICC generates an own 16-byte random number *RndB* and creates the signature as follows:

$$Sig.B = ECDSA_{Sign}(Priv.B,\ 0xF0F0[||OptsA]||RndB||RndA)$$

For *`OptsA`* the full TLV-structure is included, while for RndA and RndB only the 16-byte random values are included.

6.3.3.4 Authentication overview

The ECC-based card-unilateral authentication supported by A30 is based on the two-pass unilateral authentication as standardized in ISO/IEC 9798-3 [19] with the following modifications:

- Identities are not communicated or included in the protocol:
  - the identity of the card may be extracted from the corresponding certificate.
  - there is no requirement for knowledge and confirmation of the reader identity by the card. Note that reader's random sufficiently ensures uniqueness and timeliness, and therefore prevents the returned token to be accepted by other parties.
- The references *A* and *B* are exchanged to be more aligned with other protocols in this document.

An overview of this asymmetric card-unilateral authentication is given in Table 12.

The inclusion of a random number generated by the card prevents the reader from having full control on the data that gets "signed" by the card. This is different from a generic ECDSA signature generation as supported with CryptoRequest.

Table 12. ECC-basedcard-unilateral authentication

PCD	PICC
Knows:*Pub.B* The PCD generates a random challenge *RndA*	**Knows:***Priv.B*
OptsA || *RndA* →	
	The PICC generates a random *RndB*: The PICC computes its signature: *Sig.B = ECDSASign(Priv.B,0xF0F0*[||*OptsA*]||*RndB*||*RndA*)
RndB || *Sig.B* ←	
The PCD validates the signature: $ECDSA_{Verify}$(*Pub.B, 0xF0F0*[||*OptsA*]||*RndB*||*RndA, Sig.B*)	

6.3.4 AES-based Symmetric Authentication

6.3.4.1 Command AuthenticateEV2First

In the remainder, there is made mention of First Authentication and Non-First Authentication. A First Authentication is done in state VCState.NotAuthenticated or in one of the authenticated states, see Section 6.3.1. The Non-First Authentication can only be applied after a First Authentication, i.e. in an authenticated state. Correct application of First Authentication and Non-First Authentication allows cryptographically binding all messages within a transaction by using a transaction identifier, see Section 6.3.6.1, and a command counter, see Section 6.3.6.2, even if multiple authentications are required.

The following table specifies when to authenticate using First Authentication and when to use Non-First Authentication.

Table 13. When to use which authentication command

Purpose	First Authentication	Non-First Authentication
First symmetric authentication (i.e. when not in VCState.AuthenticatedAES)	Allowed	Not Allowed
Subsequent symmetric authentication (i.e. when in VCState.AuthenticatedAES)	Allowed, recommended not to use.	Allowed, recommended to use.

It is possible to use First Authentication when already authenticated. This can be used if the PCD does not care about interleaving attacks but rather prefers a simpler implementation. Note that the messages of the ongoing transaction are then not bound cryptographically anymore. Therefore, using First Authentication followed by Non-First Authentication is recommended. In this way, an attacker will not be able to make a PICC work with two PCDs at the same time and in that way compromise the security.

The authentication consists of two parts: AuthenticateEV2First - Part1 and AuthenticateEV2First - Part2. Detailed command definition can be found in Section 7.3.3. The protocol cannot be interrupted by other commands. On any command different from AuthenticateEV2First - Part2 received after the successful execution of the first part, the PICC aborts the ongoing authentication.

During this authentication phase, the PICC accepts messages from the PCD that are longer than the lengths derived from this specification as long as LenCap is correct. This feature is to support the upgradability features on future product versions. The current content of PCDcap2 shall not be interpreted by the PICC. The PCD rejects answers from the PICC when they don't have the proper length.

Upon reception of AuthenticateEV2First, the PICC validates the targeted key. If the key does not exist, AuthenticateEV2First is rejected. Within the application, there are 0x05 application keys available for authentication addressed by KeyNo 0x00 until 0x04. Addressing, other symmetric keys are only available for crypto operations with CryptoRequest and result in an error. At PICC level, there are no symmetric keys.

The PICC generates a random 16-byte challenge *RndB* and sends this encrypted to the PCD, according to Section 6.3.6.4. Additionally, the PICC resets CmdCtr to zero and generate a random Transaction Identifier (TI).

If the Authentication Counter is enabled for authentication counting, it shall be incremented by 1 on successful execution of AuthenticateEV2First. If the counter reaches AuthCtrLimit if enabled, any further authentication is rejected. However, once 0xFFFFFFFF is reached, the counter is not further incremented, but the authentication is still accepted.

Upon reception of the AuthenticateEV2First response from the PICC, the PCD also generates a random 16-byte challenge *RndA*. The PCD encrypts, on his turn, the concatenation of *RndA* with *RndB'*, which is the received challenge after decryption and rotating it left by one byte. Within AuthenticateEV2First - Part2, this is sent to the PICC.

Upon reception of AuthenticateEV2First - Part2, the PICC decrypts the second message and validates the received *RndB'*. If not as expected, the command is rejected. Else it generates *RndA'* by rotating left the received *RndA* by one byte. This is returned together with the generated TI. Also, the PICC sends 12 bytes of capabilities to the PCD: 6 bytes of PICC capabilities PDcap2 and 6 bytes of PCD capabilities PCDcap2 that were received on the command (sent back for verification).

If AWDT1 is enabled, see SetConfiguration, the timer is started during AuthenticateEV2First execution. If the timer expires before AuthenticateEV2First - Part2 reception, the authentication attempt is reset and the AuthenticateEV2First - Part2 will be rejected.

On successful execution of the authentication protocol, the session keys SesAuthMACKey and SesAuthENCKey are generated according to Section 6.3.4.3. The PICC is in VCState.AuthenticatedAES and the Secure Messaging is activated. On any failure during the protocol, the PICC ends up in VCState.NotAuthenticated.

If there is a mismatch between the capabilities expected by the PCD and the capabilities presented by the PICC to the PCD (both the PDcap2 and the echoed/adjusted PCDcap2), it is the responsibility of the PCD to take the proper actions based on the application the PCD is running. This decision is outside the scope of this specification.

6.3.4.2 Command AuthenticateEV2NonFirst

This section defines the Non-First authentication, which is recommended to be used if Secure Messaging is already active, see Table 13. In this procedure both, the PICC as well as the PCD show in an encrypted way that they possess the same secret, i.e. the same key. This authentication is supported with KeyType.AES128 or KeyType.AES256 keys.

The authentication consists of two parts: AuthenticateEV2NonFirst - Part1 and AuthenticateEV2NonFirst - Part2. A detailed command definition can be found in Section 7.3.4. This command is rejected if there is no active symmetric authentication. For the rest, the behavior is exactly the same as for AuthenticateEV2First, except for the following differences:

- No *PCDcap2* and *PDcap2* are exchanged and validated.
- Transaction Identifier TI is not reset and not exchanged.
- Command Counter CmdCtr is not reset.
- If the authentication Counter is enabled for authentication counting, it shall not be incremented on Section 7.3.4.

After successful authentication, the PICC remains in VCState.AuthenticatedAES. On any failure during the protocol, the PICC ends up in VCState.NotAuthenticated.

6.3.4.3 Session Key Generation

At the end of a valid authentication with AuthenticateEV2First or AuthenticateEV2NonFirst, both the PICC and the PCD generate two session keys for secure messaging, as shown in Figure 7:

- SesAuthMACKey for MACing of messages
- SesAuthENCKey for encryption and decryption of messages

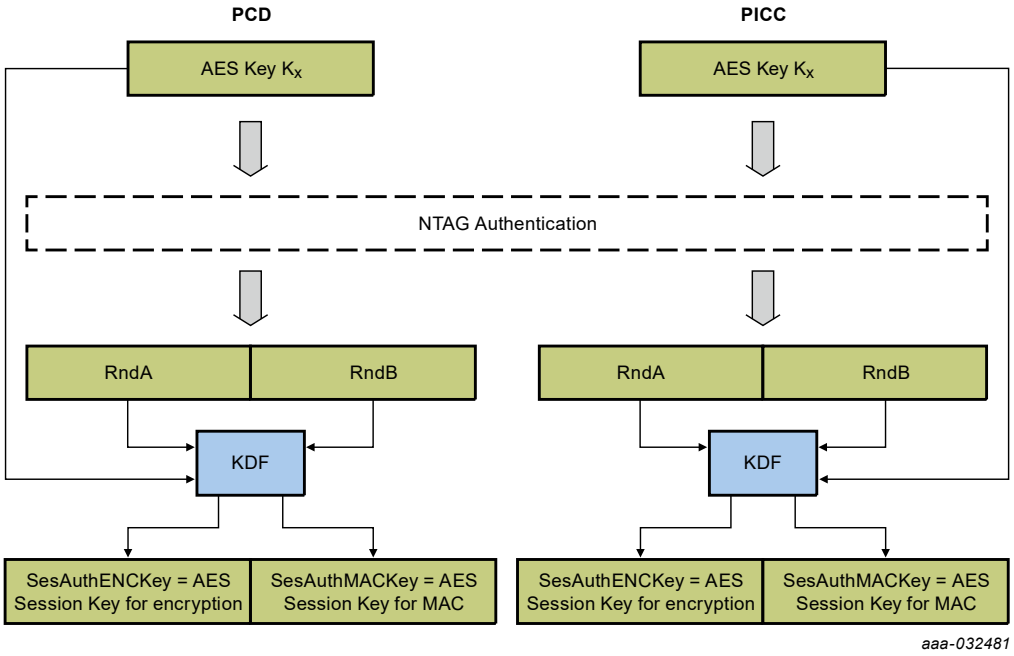


Figure 7. Session key generation for Secure Messaging

The session key generation is according to NIST SP 800-108 [9] in counter mode.

The Pseudo Random Function PRF(key; message) applied during the key generation is the CMAC algorithm described in NIST Special Publication 800-38b [7]. The key derivation key is the key Kx that was applied during authentication. If the authentication targets a KeyType.AES128 key, the generated session keys are also of KeyType.AES128. If a KeyType.AES256 authentication key is targeted, the session keys are also KeyType.AES256.

The input data is constructed using the following fields as defined by [9]. NIST SP 800-108 allows defining a different order than proposed by the standard as long as it is unambiguously defined.

- a 2-byte label, distinguishing the purpose of the key: 0x5AA5 for MACing and 0xA55A for encryption
- a 2-byte counter
  - KeyType.AES128: fixed to 0x0001.
  - KeyType.AES256: counting from 0x0001 to 0x0002.
- a 2-byte length,
  - KeyType.AES128: fixed to 0x0080.
  - KeyType.AES256: fixed to 0x0100.
- a 26-byte context, constructed using the two random numbers exchanged, RndA and RndB

KeyType.AES128

First, the 32-byte input session vectors SV_x are derived as follows [1] :

$SV1 = A5h||5Ah||00h||01h||00h||80h||RndA[15..14]||(\text{RndA}[13..8] \oplus RndB[15..10])||RndB[9..0]||RndA[7..0]$
$SV2 = 5Ah||A5h||00h||01h||00h||80h||RndA[15..14]||(\text{RndA}[13..8] \oplus RndB[15..10])||RndB[9..0]||RndA[7..0]$

with $\oplus$ being the XOR-operator.

Then, the 16-byte session keys are constructed as follows:

$SesAuthENCKey = PRF(K_x, SV1)$

1 Bytes are numbered from rightmost to leftmost i.e. index 0 for the rightmost byte.

SesAuthMACKey = PRF(K_x, SV2)

KeyType.AES256

First, the 32-byte input session vectors SV x are derived as follows:

SV1a = 0xA5||0x5A||0x00||0x01||0x01||0x00||RndA[15..14]||(RndA[13..8] ⊕ RndB[15..10])||RndB[9..0]||RndA[7..0]
SV1b = 0xA5||0x5A||0x00||0x02||0x01||0x00||RndA[15..14]||(RndA[13..8] ⊕ RndB[15..10])||RndB[9..0]||RndA[7..0]
SV2a = 0x5A||0xA5||0x00||0x01||0x01||0x00||RndA[15..14]||(RndA[13..8] ⊕ RndB[15..10])||RndB[9..0]||RndA[7..0]
SV2b = 0x5A||0xA5||0x00||0x02||0x01||0x00||RndA[15..14]||(RndA[13..8] ⊕ RndB[15..10])||RndB[9..0]||RndA[7..0]

with ⊕ being the XOR-operator.

Then, the 32-byte session keys are constructed as follows:

SesAuthENCKey = PRF(K_x, SV1a)||PRF(Kx, SV 1b)
SesAuthMACKey = PRF(K_x, SV2a)||PRF(Kx, SV 2b)

6.3.5 AuthenticationCounter and Limit

To allow mitigating potential future attack scenarios, symmetric mutual authentications can be configured with a counter and usage limitation. This allows limiting the amount of key computations, and therefore related trace collection for side-channel attacks. Next to attack mitigation, this feature can also be used to limit the usage of a card/device. Potentially, the limit can be increased in the field, e.g. if the end user pays for additional service.

The authentication counter and usage limitation are configured through SetConfiguration Option 0x16, by assigning one of the FileType.Counters to this purpose.

Once enabled, A30 shall maintain a AuthCtr through the assigned file, for counting the authentications, and if configured also an AuthCtrLimit.

This means that the AuthCtr shall be incremented by the following operations, if enabled:

- AuthenticateEV2First for AES-based authentication, before the response of the Part 1.

If the configured AuthCtrLimit has been reached, the related authentication is disabled. This means that the relevant keys cannot be used anymore, though the key entry can still be updated (and potentially reenabled) if the required authentication to do so can still be gained, e.g. through an asymmetric authentication if symmetric authentication is disabled.

If the AuthCtrLimit is disabled, authentications may still be counted.

When further updating SetConfiguration Option 0x16, it is possible to disable or change the AuthCtrLimit without affecting the current AuthCtr value. This ensures the monotonic property of the FileType.Counter. When configuring a different file, the authentication counting for the original file is disabled. Putting the limit to a value equal or lower than the current value will immediately disable the authentication.

As any other SetConfiguration option, the current authentication counter configuration and AuthCtrLimit can be retrieved with GetConfiguration. For this Option 0x16, also the current AuthCtr value will be returned by GetConfiguration.

Enabling the feature may create a denial-of-service risk. It must be assessed from a system-level perspective if this can be accepted.

6.3.6 EV2/AES secure messaging

The EV2 secure messaging is an AES-based secure messaging, which was introduced in MIFARE DESFire EV2, explaining the naming.

The EV2 secure messaging can both be initiated by an ECC-based mutual authentication as defined in Section 6.3.2, as well as by the AES-based mutual authentication as defined in Section 6.3.4.

6.3.6.1 Transaction Identifier

To avoid interleaving of transactions from multiple PCDs toward one PICC, the Transaction Identifier (TI) is included in each MAC that is calculated over commands or responses. The TI is generated by the PICC and communicated to the PCD with a successful execution of an AuthenticateEV2First command, see Section 7.3.3. The size is 4 bytes and these 4 bytes can hold any value. The TI is treated as a byte array, so there is no notion of MSB and LSB.

In the case of ECC-based authentication it is expected that a transaction only consists of a single authentication. As a CARootKey and/or reader certificate can cover multiple access rights, see Section 6.4, there should not be a need to authenticate multiple times. Therefore, in VCState.AuthenticatedECC, the TI is set to all zero bytes.

6.3.6.2 Command Counter

A command counter is included in the MAC calculation for commands and responses to prevent e.g. replay attacks. It is also used to construct the Initialization Vector (IV) for encryption and decryption.

Each command, besides few exceptions, see below, is counted by the command counter CmdCtr, which is a 16-bit unsigned integer. Both sides count commands, so the actual value of the CmdCtr is never transmitted. The CmdCtr is reset to 0x0000 at PCD and PICC after a successful AuthenticateEV2First authentication and it is maintained as long as the PICC remains authenticated. In cryptographic calculations, the CmdCtr is represented LSB first. Subsequent authentications using AuthenticateEV2NonFirst do not affect the CmdCtr. Subsequent authentications using the AuthenticateEV2First will reset the CmdCtr to 0x0000.

In the case of ECC-based authentication, the CmdCtr is also set to 0x0000 after successful authentication, i.e. a ISOGeneralAuthenticate exchange successfully completing SIGMA-I mutual authentication.

The CmdCtr is increased between the command and response, for all communication modes.

For CommMode.Plain, this is not reflected in the actual command exchange as the CmdCtr is not used.

When a MAC on a command is calculated at PCD side that includes the CmdCtr, it uses the current CmdCtr. The CmdCtr is afterward incremented by 1. At PICC side, a MAC appended to received commands is checked using the current value of CmdCtr. If the MAC matches, CmdCtr is incremented by 1 after successful reception of the command, and before sending a response.

For CommMode.Full, the same holds for both the MAC and encryption IV calculation, i.e. the nonincreased value is used for the command calculations while the increased value is used for the response calculations.

If the CmdCtr holds the value 0xFFFF and a command maintaining the active authentication arrives at the PICC. This leads to an error response and the command is handled like the MAC was wrong.

Command chaining, see Section 6.2.3, does not affect the counter. The chained command is considered as a single command, just as for the other aspects of secure messaging, and therefore the related counter is increased only once.

6.3.6.3 MAC Calculation

MACs are calculated using the underlying block cipher according to the CMAC standard described in [7]. Padding is applied according to the standard.

The MAC used in A30 is truncated by using only the 8 even-numbered bytes out of the 16-bytes output as described [7] when represented in most-to-least-significant order.

Initialization vector for MACing

The initialization vector used for the CMAC computation is the zero-byte IV as prescribed [7].

6.3.6.4 Encryption

Encryption and decryption are calculated using AES according to the CBC mode of NIST SP800-38a [6].

Padding is applied according to Padding Method 2 of ISO/IEC 9797-1 [8], i.e. by always adding 0x80 followed. If required, by zero bytes until a string with a length of a multiple of 16 byte is obtained. If the plain data is a multiple of 16 bytes already, an additional padding block is added. The only exception is during the authentication itself (AuthenticateEV2First and AuthenticateEV2NonFirst), where no padding is applied at all.

The notation *E(key, message)* is used to denote the encryption and *D(key, message)* for decryption.

Initialization Vector for Encryption

When encryption is applied to the data sent between the PCD and the PICC, the Initialization Vector (IV) is constructed by encrypting with SesAuthENCKey according to the ECB mode of NIST SP800-38a [6] the concatenation of:

- a 2-byte label, distinguishing the purpose of the IV: 0xA55A for commands and 0x5AA5 for responses
- Transaction Identifier TI
- Command Counter CmdCtr (LSB first)
- Padding of zeros acc. to NIST SP800-38b [7]

This results in the following IVs:

IV for CmdData = *E(SesAuthENCKey; 0xA5* || *0x5A* || TI || CmdCtr || *0x0000000000000000)*

IV for RespData = *E(SesAuthENCKey;5Ah* || *0xA5* || TI || CmdCtr || *0x0000000000000000)*

When an encryption or decryption is calculated, the CmdCtr to be used in the IV are the current values. This means that if CmdCtr = *n* before the reception of a command, after the validation of the command CmdCtr = *n* + *1* and that value will be used in the IV for the encryption of the response.

For the encryption during authentication (both AuthenticateEV2First and AuthenticateEV2NonFirst), the IV is 128 bits of 0.

6.3.6.5 Session Key Generation

As an output of a successful authentication, both the PICC and the PCD have generated two session keys for secure messaging:

- SesAuthMACKey or *KSesAuthMAC* for MACing of messages
- SesAuthENCKey or *KSesAuthENC* for encryption and decryption of messages

These session keys are generated differently, depending on the authentication:

- For ECC-based authentication, this is defined in Section 6.3.2.
- For AES-based authentication, this is defined in Section 6.3.4.

6.3.6.6 Communication Modes

A30 supports three communication modes as defined in Table 14. As shown in the table, the different communication modes are represented by two bits. This representation is used at several places in the document.

Table 14. Supported communication modes

Communication Mode	Bit Representation	Explanation
CommMode.Plain	X0	No protection: message is transmitted in clear
CommMode.MAC	01	MAC protection for integrity and authenticity
CommMode.Full	11	Full protection for integrity, authenticity, and confidentiality

The communication mode defines the level of security for the communication as further explained in the next subsections.

6.3.6.7 Plain Communication Mode

The command and response data is not secured. The data is sent in plain, see Figure 8, i.e. as defined in the command specification tables, see Section 7.

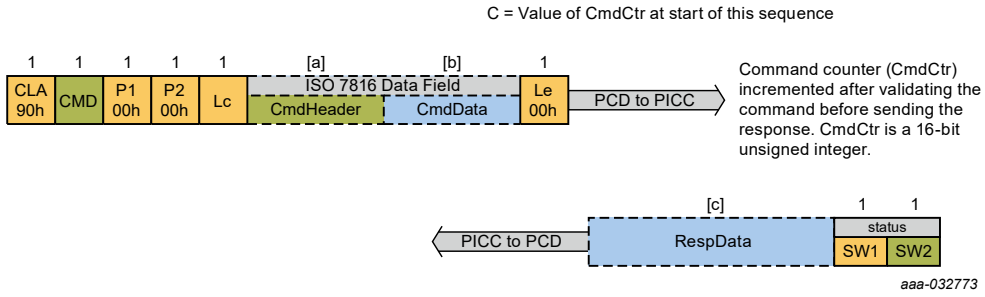


Figure 8. Plain Communication Mode

However, as the PICC is in authenticated state, the command counter CmdCtr is still increased as defined in Section 6.3.6.2.

This allows the PCD and PICC to detect any insertion and/or deletion of commands sent in CommMode.Plain on any subsequent command that is sent in CommMode.MAC or CommMode.Full.

6.3.6.8 MAC Communication Mode

The Secure Messaging applies MAC to all commands listed as such in Section 7.2.

In the case MAC is to be applied, the following holds. The MAC is calculated using the current session key SesAuthMACKey. MAC calculation is done as defined in Section 6.3.6.3.

For commands, the MAC is calculated over the following data (according to the definitions from Section 6.2.1) in this order:

- Cmd
- Command Counter CmdCtr
- Transaction Identifier TI
- Command header - CmdHeader (if present)
- Command data - CmdData (if present)

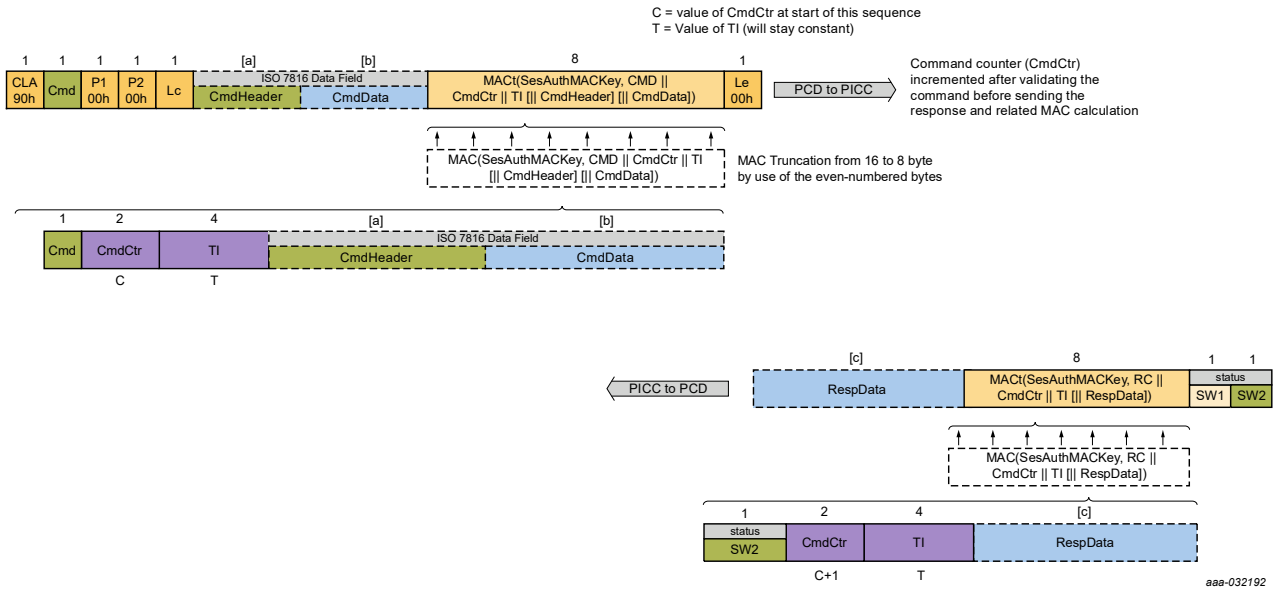
For responses, the MAC is calculated over the following data in this order:

- Return code - RC
- Command Counter -CmdCtr (The already increased value)
- Transaction Identifier -TI
- Response data - RespData (if present)

CmdCtr is the Command Counter as defined in Section 6.3.6.2. The CmdCtr is increased between the computation of the MAC on the command and the MAC on the response. TI is the Transaction Identifier, as defined in Section 6.3.6.1. The other input parameters are as defined in Section 6.2.1. The calculation is illustrated in Figure 9.

In case of command chaining, the MAC calculation is not interrupted. The MAC is calculated over the data including the complete data field (i.e. either CmdData or RespData of all frames) at once. The MAC is always transmitted by appending to the unpadded plain command. If necessary, an additional frame is sent. If a MAC over the command is received, the PICC verifies the MAC and rejects commands that do not contain a valid MAC by returning INTEGRITY_ERROR.

In this case, the ongoing command and transaction are aborted (see also Section 7). The authentication state is immediately lost and the error return code is sent without a MAC appended. Any other error during the command execution has the same consequences.



CLA, P1, P2, LC, Le, and SW1 are not included in secure messaging calculation. SW2 is the return code (RC) and appended in the beginning for secure messaging calculation.

Figure 9. Secure Messaging: MAC Communication mode

6.3.6.9 Full Communication Mode

The Secure Messaging applies encryption (CommMode.Full) to all commands listed as such in Section 7.2. In the case CommMode.Full is to be applied, the following holds. The encryption/decryption is calculated using the current session key SesAuthENCKey. Calculation is done as defined in Section 6.3.6.4 over either the command or the response data field (i.e. CmdData or RespData). None of the commands have a data field in both the command and the response frame.

After the encryption, the command and response frames are handled as with MAC. This means that additionally a MAC is calculated and appended for transmission using the current session key SesAuthMACKey. This is exactly done as specified for MAC in Section 6.3.6.8, replacing the plain CmdData or RespData by the

encrypted field: *E(SesAuthENCKey; CmdData)* or *E(SesAuthENCKey; RespData)*. The complete calculation is illustrated in Figure 10. In the case of command chaining, the encryption/decryption is applied over the complete data field (i.e. of all frames). If necessary, due to the padding or the MAC added, an additional frame is sent. If encryption of the command is required, after the MAC verification as described for MAC, the PICC verifies and removes the padding bytes. Commands without a valid padding are also rejected by returning INTEGRITY_ERROR.

In this case, the ongoing command and transaction are aborted (see also Section 7). The authentication state is immediately lost and the error return code is sent without a MAC appended. Any other error during the command execution has the same consequences.

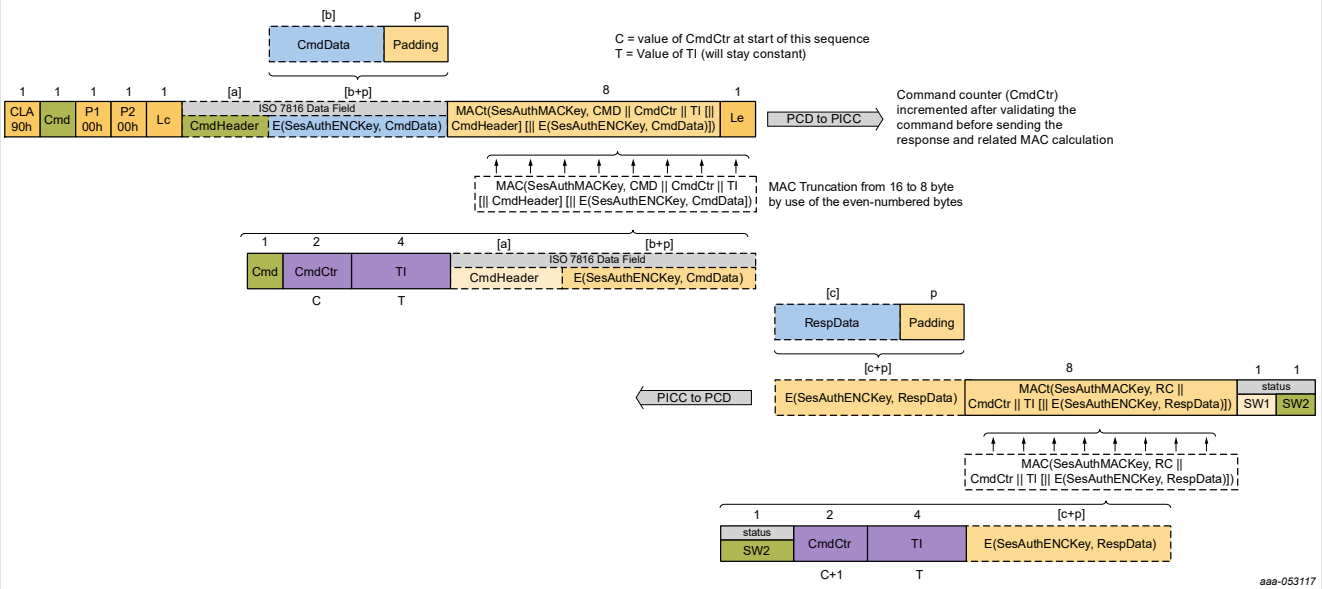


Figure 10. Secure Messaging: CommMode.Full

6.3.7 Controller Session Key Usage

As described in Section 6.3.2, A30 supports both the initiator and responder roles of SIGMA-I for the ECC-based mutual authentication. This may open up use case where A30 is used in both the verifier (controller in I^2C) and prover device (target in I^2C). For example, this can be a host device and consumable parts.

For these use cases, A30 supports ProcessSM, to allow command generation, i.e. applying the required secure messaging, and response processing, i.e. validating the MAC and eventually decrypting.

Support of ProcessSM must be explicitly enabled through SetConfiguration Option 0x0F and I^2C interfaces respectively, by setting bit 7 of the ProtocolOptions. By default, ProcessSM is disabled for both interfaces. The typical use case requires this only over the I^2C interface.

6.3.7.1 ProcessSM

If ProcessSM is enabled, it is only supported in VCState.AuthenticatedECC (see Section 6.3.1). In other states, the command is rejected.

The ProcessSM supports two variants:

- ProcessSM_Apply: applying secure messaging to a command.
- ProcessSM_Remove: removing secure messaging from a response.

The overall command format is outlined in Section 7.3.5 while the two variants are further detailed in the following subsections.

While the ProcessSM is issued in VCState.AuthenticatedECC, the command itself is not protected by the regular secure messaging. This means both the command and response are issued in plain, just applying the secure messaging to the command and response data to support controller device processing.

Once enabled, there is no additional access control to the ProcessSM command. Therefore, it must be assessed at system level if the access to the command can be abused.

6.3.7.2 ProcessSM_Apply

The ProcessSM_Apply is used for applying secure messaging to a command before sending it to the target device. The command format is outlined in Section 7.3.6.

If targeting CommMode.Plain, there is no command data exchanged.A30 increments the CmdCtr by the amount given in CmdCtrIncr. If CmdCtr would reach 0xFFFF or overflow, the command is rejected.

If targeting CommMode.MAC or CommMode.Full, Plaintext provides the data to be protected. ProcessSM_Apply only supports one-shot operations fitting in a single short-length ISO/IEC 7816-4 APDU. Bigger lengths are rejected. The data provided and returned by A30 does not hold the ISO/IEC 7816-4 APDU wrapping overhead. This means the native command fields, as described in Section 6.2.1 consisting of Cmd (i.e. the ISO/IEC 7816-4 INS field) followed by eventually CmdHeader and CmdData fields (i.e.the full ISO/IEC 7816-4 Command Data field).

If targeting CommMode.Full, an additional parameter Offset indicates where the encryption shall start, i.e.the first byte of CmdData.

If targeting CommMode.MAC, only the computed MAC is returned. If targeting CommMode.Full, the encrypted data is returned together with the computed MAC. Other plain data like the Cmd and CmdHeader are not echoed.

6.3.7.3 ProcessSM_Remove

The ProcessSM_Remove is used for removing and validating secure messaging from a response received from a target device. The command format is outlined in Section 7.3.7.

If targeting CommMode.Plain, the command must not be called as the only relevant processing (CmdCtr), is triggered with ProcessSM_Apply.

If targeting CommMode.MAC or CommMode.Full, Ciphertext provides the data to be processed. Note that ProcessSM_Remove only supports one-shot operations fitting in a single short-length ISO/IEC 7816-4 APDU. Bigger lengths are rejected. The data provided and returned by A30 does not hold the ISO/IEC 7816-4 APDU wrapping overhead. This means the native response fields, as described in Section 6.2.1 consisting of RC (i.e. the ISO/IEC 7816-4 SW2 field) followed by eventually RespData and the MAC (i.e. the full received ISO/IEC 7816-4 Response Data field). In CommMode.Full, RespData is the encrypted response data.

If targeting CommMode.MAC, no data is returned. If targeting CommMode.Full, the decrypted data is returned. The RC is not echoed.

6.3.8 Secure Dynamic Messaging

The Secure Dynamic Messaging (SDM) allows for confidential and integrity-protected data exchange, without requiring a preceding authentication. A30 supports SDM for reading from one of the StandardData files on the PICC. Secure Dynamic Messaging allows adding security to the data read, while still being able to access it with standard NDEF readers. The typical use case is an NDEF holding a URI and some metadata, where SDM allows this metadata to be communicated confidentiality and integrity protected toward a back end server.

When using SDM, residual risks coming with the Secure Dynamic Messaging for Reading have to be taken into account. As SDM allows free reading of the secured message, i.e. without any up-front reader authentication, anybody can read out the message. This means that also a potential attacker is able to read out and store one ore multiple messages, and play them at a later point in time to the verifier.

If this residual risk is not acceptable for the system's use case, one of the authentication protocols (using challenge response protocol) and subsequent secure messaging should be applied. This would require using an own application and operating outside a standard NDEF read operation.

Other risk mitigation may be applied for SDM to limit the residual risk, without completely removing it:

- Track SDMReadCtr per tag at the verifying side. Reject SDMReadCtr values that have been seen before or that are played out-of-order. This is a minimum requirement that any verifier should implement.
- Limit the time window of an attacker by requiring tags to be presented regularly (e.g. at least once a day) in combination with the previous mitigation.
- Read out the SDM-protected file more than once. This does not protect against attackers that have read out the valid tag also multiple times and play the received responses in the same sequence. A30 supports two modes for integrity protection and authentication of the data:

A30 supports two modes for integrity protection and authentication of the data:

- symmetric SDMMAC, where the data is protected by a Message Authentication Code, which is generated by a symmetric AES key. This is specified in Section 6.3.8.8 and Section 6.3.8.9.
- asymmetric SDMSIG, where the data is protected by a Signature, which is generated with the private key of an ECC key pair. This is specified in Section 6.3.8.10 and Section 6.3.8.11. This approach may ease the key management towards e.g. reader infrastructure as no secret key is required for the signature validation.

Encryption is always based on symmetric cryptography, as specified in Section 6.3.8.4 for PICCData like the UID and Section 6.3.8.7 for generic file data (SDMENCFileData).

The session key derivation for symmetric keys (be it for encryption or MACing) is outlined in Section 6.3.8.12.

SDM is enabled and configured with ChangeFileSettings, see Section 6.10.2.3. Access right related aspects are defined in Section 6.10.2.1.

6.3.8.1 SDM Read Counter

To allow replay detection by the party validating the data read, a read counter is associated with the file for which Secure Dynamic Messaging is enabled.

SDMReadCtr is a 24-bit unsigned integer. The SDMReadCtr is reset to 0x000000 when enabling SDM with ChangeFileSettings. In cryptographic calculations and represented with binary encoding on the external interface, the SDMReadCtr is represented LSB first. When represented with ASCII encoding on the contactless interface, it is represented MSB first. This is in line with the NFC counter representation in [13].

In not Authenticated state, the SDMReadCtr is incremented by 1 before calculating the response of the first read command, ReadData or ISOReadBinary, if successful. On subsequent read commands targeting the same file, the SDMReadCtr is not increased, and the current value is used. As soon as a different command has been received, the counter is incremented again on a subsequent read command. Also when varying between ReadData and ISOReadBinary, the counter is incremented on each first instance of the read command type. The SDMReadCtr is not incremented when authenticated.

If the SDMReadCtr reaches the SDMReadCtrLimit (see Section 6.3.8.2) or the value 0xFFFFFF (if SDMReadCtrLimit is not enabled) and a first read command arrives at the PICC, an error is being returned. Command chaining, see Section 6.2.3, does not additionally affect the counter increase. The chained command is considered as a single command.

SDMReadCtr can be retrieved via the mirroring as part of the PICCData, see Section 6.3.8.3, or it can be retrieved via GetFileCounters.

6.3.8.2 SDM Read Counter Limit

To allow limiting the number of reads that can be done with a single device applying Secure Dynamic Messaging, an optional SDM Read Counter Limit can be configured. There are two main use cases:

- Limit the number of usages from the card side. Typically this can also be controlled from the back end verifying the SDM for Read protected message.
- Limit the number of traces that can be collected on the symmetric crypto processing. This way the attack potential via side-channel attacks can be further reduced.

The number of reads that can be executed for an SDM configured file can be limited by setting an SDM Read Counter Limit (SDMReadCtrLimit). This is an unsigned integer of 3 bytes, related with SDMReadCtr. On the interface, the SDMReadCtrLimit is represented LSB first. The SDMReadCtrLimit can be enabled by setting a customized value with ChangeFileSettings. It can be retrieved with GetFileSettings.

Once the SDMReadCtr equals the SDMReadCtrLimit, no reading of the file with ReadData or ISOReadBinary in not authenticated state can be executed. If authenticated, reading is always possible even if SDMReadCtrLimit is reached, applying the regular secure messaging. If the SDMReadCtrLimit is disabled with ChangeFileSettings, this is also equivalent to putting it to the maximum value: 0xFFFFFF.

6.3.8.3 PICCData

The PICCData holds metadata of the targeted PICC and file, consisting of the UID and/or the SDMReadCtr. Whether PICCData is transmitted in plain or encrypted depends on the configuration of the SDMMetaRead access rights on the file, see Section 7. If the SDMMetaRead access right is configured for free access (0xE), PICCData is plain and is defined according to Table 15.

ASCII mirroring is reflected by the function EncodeASCII(), which means that each hexadecimal character of the hexadecimal representation will be ASCII encoded using capitals. For example, the UID 0x04E141124C2880 becomes: 0x30 34h 0x45 31h 0x34 31h 0x31 32h 0x34 43h 0x32 38h 0x38 30h.

Table 15. PICCData: plain encoding and lengths

Mode	PICCData Value	Length with 7-byte UID
ASCII	EncodeASCII(UID)	UIDLength = 14 (i.e. 2*UIDLen)
ASCII	EncodeASCII(SDMReadCtr)	SDMReadCtrLength = 6 (i.e. 2 × 3)

The SDMReadCtr, as defined in Section 6.3.8.1, is represented MSB first for the ASCII case. If the SDMMetaRead access right is configured for an application key, PICCData is encrypted as defined in Section 6.3.8.4. In this case, the input plaintext for the encryption is always in binary encoding, while the output ciphertext will be ASCII encoded.

The PICCData is mirrored within the file. This is configured with ChangeFileSettings via the related offsets.

In the case of plain mirroring (i.e. access right SDMMetaRead = 0xE):

- UIDOffset configures the UID mirroring position. It is only given if UID mirroring is enabled.
- SDMReadCtrOffset configures the SDMReadCtr mirroring position. It is only given if SDMReadCtr mirroring is enabled. It is possible to enable the SDMReadCtr but without mirroring by putting SDMReadCtrOffset to 0xFFFFFF. In this case, it can be retrieved with the GetFileCounters command.

If UID and SDMReadCtr are mirrored within the file, they shall not overlap:

- UIDOffset ≥ SDMReadCtrOffset + SDMReadCtrLength OR SDMReadCtrOffset ≥ UIDOffset + UIDLength.

In the case of encrypted mirroring (i.e. SDMMetaRead = 0x0..0x4), PICCDataOffset configures the PICCData mirroring. The encryption is outlined in Section 6.3.8.4.

If the PICCData is mirrored within the file, the mirroring shall always be applied in not authenticated state, independently of whether Secure Dynamic Messaging applies. This means it will also be applied if reading the file with free access due to Read or ReadWrite access right. If authenticated, no mirroring is done, i.e. the regular secure messaging is always applied on the static file data.

With A30, PICCData is always ASCII encoded.

When both the UID and SDMReadCtr are mirrored, “x” (0x78) is used as a separator character. This can be achieved by leaving one byte space between the placeholders defined by UIDOffset and SDMReadCtrOffset, and writing “x” (0x78) in the static file data.

6.3.8.4 Encryption of PICCData

In the case of encrypted PICCData mirroring (both binary and ASCII), PICCDataTag specifies what metadata is mirrored, together with the length of the UID if mirrored, as defined in Table 16.

Table 16. PICCDataTag

Bit	Value	Description
Bit 7	-	UID mirroring
	0	disabled
	1	enabled
Bit 6	-	SDMReadCtr mirroring
	0	disabled
	1	enabled
Bit 5-4	00	RFU
Bit 3-0	-	UID Length
	0x0	RFU (if UID is not mirrored)
	0x7	7 byte UID

The format of the plain text is: *PICCDataTag [|| UID] [|| SDMReadCtr].*

To ensure that the encrypted PICCData cannot be abused for tracking purposes, random padding is added to the actual plain text input.

The random padding is generated for the response of the first read command, ReadData or ISOReadBinary. On subsequent read commands targeting, the same file the same random padding is reused. This allows for reading the file in chunks, where a chunk border might even be in the middle of the encrypted PICCData. As soon as a different command has been received, fresh random padding is generated on a subsequent read command. Also when varying between ReadData and ISOReadBinary, fresh random padding is generated.

The key applied for encryption of PICCData is the SDMMetaReadKey as defined by the SDMMetaRead access right.

6.3.8.4.1 AES mode encryption

Encryption and decryption of the PICCData are calculated using the underlying block cipher according to the CBC mode of NIST SP800-38a [6], applying zero-byte IV.

A30 supports AES-128 and AES-256 as the underlying block cipher depending on the key type of the SDMMetaReadKey.

Therefore, PICCData is defined as follows:

PICCData = E(SDMMetaReadKey; PICCDataTag [|| UID] [|| SDMReadCtr] || RandomPadding) with PICCDataTag as defined in Section 6.3.8.3, and RandomPadding being a random byte string generated by the PICC to make the input 16 bytes long. Because of the ASCII encoding, the required placeholder length doubles.

6.3.8.5 GPIOStatus

When one of the GPIO pins is configured for input, see Section 6.13, or tag tamper detection, see Tag Tamper Protection, with SetConfiguration 0x11, see Section 6.3.8.5, it is possible to mirror the statuses within the NDEF file.

The GPIO statuses are encoded on a 3-byte string, identical as the ReadGPIO response, see Section 6.10.2.3 and especially Table 254.

They can be mirrored in plain or encrypted. For the latter, GPIOStatus needs to be positioned within the placeholder for the plain data that serves as input for SDMENCFileData, see Section 6.3.8.6. In this case, the static file data is replaced by the dynamic statuses before applying the encryption. Note however, that either all status bytes are plain, or all are encrypted.

As the status bytes are already ASCII encoded, no ASCII encoding must be applied on top, and only a 3-byte placeholder is required. Where the status is mirrored within the file, is configured with ChangeFileSettings, see Section 6.10.2.3 via GPIOStatus. The restrictions on this offset shall be that it may not be overlapped with any PICCData mirrored or with the SDMMAC.

If the GPIOStatus is mirrored within the file, the mirroring shall always be applied in VCState.NotAuthenticated, independently of whether Secure Dynamic Messaging applies. This means it will also be applied if reading the file with free access due to FileAR.Read or FileAR.ReadWrite. If authenticated, i.e. in VCState.AuthenticatedAES or VCState.AuthenticatedECC, no mirroring is done, i.e. the regular secure messaging is always applied on the static file data.

6.3.8.6 SDMENCFileData

SDM for Reading supports mirroring (part of the) file data encrypted. This part is called the SDMENCFileData.

If the SDMFileRead access right is configured for an application key, part of the file data can optionally be encrypted as defined in Section 6.3.8.7 when being read out in not authenticated state.

In this case, the input plaintext for the encryption is always in binary encoding, while the output ciphertext is ASCII encoded.

If authenticated, no Secure Dynamic Messaging is applied, i.e. the regular secure messaging is always applied on the static file data.

The SDMENCFileData (if any) is always mirrored within the file. This is configured with ChangeFileSettings, see Section 7.8.7 via SDMENCOffset and SDMENCLength. If the SDMFileRead access right is disabling Secure Dynamic Messaging for reading (i.e. set to 0xF), SDMENCOffset and SDMENCLength are not present in ChangeFileSettings.

If PICCData is mirrored within the file, SDMENCFileData shall not overlap with it. Depending on what is exactly mirrored, the following holds:

- *SDMENCOffset ≥ PICCDataOffset + PICCDataLength OR PICCDataOffset ≥ SDMENCOffset + SDMENCLength.*
- *SDMENCOffset ≥ UIDOffset + UIDLength OR UIDOffset ≥ SDMENCOffset + SDMENCLength.*
- *SDMENCOffset ≥ SDMReadCtrOffset + SDMReadCtrLength OR SDMReadCtrOffset ≥ SDMENCOffset + SDMENCLength.*

It is ensured that SDMENCOffset + SDMENCLength is smaller than or equal to the file size. As the SDMMAC is as well mirrored into the file, additional conditions apply, see Section 6.3.8.8. The SDMENCLength is a multiple of 32 bytes for the ASCII encoding. With A30, only ASCII encoding is supported.

6.3.8.7 Encryption of SDMENCFileData

The key applied for the encryption is a session key SesSDMFileReadENCKey derived from the application key defined by the SDMFileRead access right as specified in Section 6.3.8.12.

From the user point of view, the SDMENCOffset and SDMENCLength define a placeholder within the file where the plain data is to be stored when writing the file.

For ASCII encoding, only the first half of the placeholder is used for storing the plain data. The second half is ignored for constructing the returned data when reading with SDM. For example, if targeting to encrypt 2 AES blocks, i.e. 32 bytes, a placeholder of 64 bytes is reserved via SDMENCOffset and SDMENCLength. The first 32 bytes hold the plaintext, and the next 32 bytes are ignored when reading with Secure Dynamic Messaging.

6.3.8.7.1 AES mode encryption

Encryption and decryption of the SDMENCFileData are calculated using the underlying block cipher according to the CBC mode of NIST SP800-38a [6].

A30 supports AES-128 and AES-256 as the underlying block cipher depending on the key type of the SDMFileReadKey.

The following IV is applied:

IV = E(SesSDMFileReadENCKey; SDMReadCtr||0x00000000000000000000000000) with SDMReadCtr LSB first.

For applying SDM with ASCII encoding, the SDMENCFileData is defined as follows:

SDMENCFileData = E(SesSDMFileReadENCKey; StaticFileData[SDMENCOffset:: SDMENCOffset + SDMENCLength=2 - 1]) with StaticFileData being the current file data as written in the placeholder. The file configuration ensures via SDMENCLength that the input is a multiple of 16 bytes, so no padding is applied.

It is possible via the read command parameters to read-only part of the file. If the SDMENCFileData is partially read as per the issued offset and length, a truncated part of the ciphertext is returned. As truncation might happen in the middle of an AES block. This means subsequent read commands to fetch the remainder of the file might be required to be able to decrypt.

6.3.8.8 SDMMAC

SDM for Reading supports calculating a MAC over the response data. This message authentication code is called the SDMMAC.

If FileAR.SDMFileRead is configured for an application key, and FileAR.SDMFileRead2 is set to 0xF, a MAC is calculated as defined in Section 6.3.8.9 when being read out in no authenticated state.

The SDMMAC is to be mirrored within the file via SDMMACOffset. This is configured with ChangeFileSettings, see Section 7.8.7.

If SDMMAC is mirrored within the file, it is limited to start only after SDMENCFileData, i.e. SDMMACOffset ≥ SDMENCOffset + SDMENCLength. The SDMMACInputOffset must ensure that the complete SDMENCFileData is included in the MAC calculation.

As the mirrored SDMMAC is ASCII encoded, the output size doubles to 16 bytes.

It is ensured that SDMMACOffset + SDMMACLength is smaller or equal than the file size. If authenticated, no Secure Dynamic Messaging is applied and the placeholder data at SDMMACOffset is not replaced, i.e. the regular secure messaging is always applied on the static file data.

The SDMMACInputOffset defines from which position in the file the MAC calculation starts. If SDMMAC is mirrored within the file, SDMMACInputOffset must be smaller than or equal to SDMMACOffset.

MACing is mandatory if the SDMFileRead access right is configured for an application key. If the SDMFileRead access right is disabling Secure Dynamic Messaging for reading (i.e. set to 0xF), SDMMACOffset and SDMMACInputOffset are not present in ChangeFileSettings.

With A30, only ASCII encoding is supported. SDMMAC is always mirrored within the file.

6.3.8.9 MAC Calculation

The key applied for the MAC calculation is a session key SesSDMFileReadMACKey derived from the application key defined by the SDMFileRead access right, as specified in Section 6.3.8.12.

6.3.8.9.1 AES mode MAC calculation

The 8-byte SDMMAC is calculated using AES according to the CMAC standard described in NIST Special Publication 800-38b [7] applying the same truncation as the AES mode secure messaging, see Section 6.3.6.3.

A30 supports AES-128 and AES-256 as the underlying block cipher depending on the key type of the SDMFileReadKey.

The SDMMAC is defined as follows:

SDMMAC = MACt (SesSDMFileReadMACKey; DynamicFileData[SDMMACInputOffset ... SDMMACOffset - 1]) with DynamicFileData being the file data as how it is put on the contactless interface, i.e. replacing any placeholders by the dynamic data.

6.3.8.10 SDMSIG

If FileAR.SDMFileRead2 is configured for an application ECCPrivateKey, a signature, called SDMSIG, is calculated as defined in Section 6.3.8.11 when being read out in VCState.NotAuthenticated. If the targeted ECCPrivateKey does not exist or is not enabled for ECC-based Secure Dynamic Messaging (via its key policy or if KeyUsageCtrLimit was already reached, see Section 6.7.1.2), the read command is rejected.

The offsets for signature input and signature mirroring are configured with ChangeFileSettings, see Section 6.10.2.3. As to a large extent the same rules apply, parameters SDMMACOffset and SDMMACInputOffset are reused.

This means that the SDMSIG is to be mirrored within the file via SDMMACOffset. It shall be limited to start only after SDMENCFileData, i.e. *SDMMACOffset* ≥*SDMENCOffset* +*SDMENCLength*. The SDMMACInputOffset must ensure that the complete SDMENCFileData is included in the signature calculation. The SDMSIGLength is128 bytes, as only ASCII encoding is supported. It shall be ensured that *SDMMACOffset + SDMSIGLength* is smaller or equal than the file size.

Also here, if authenticated, no Secure Dynamic Messaging is applied and the placeholder data at SDMMACOffset is not replaced, i.e. the regular secure messaging is always applied on the static file data.

The SDMMACInputOffset defines from which position in the file the input for the signature calculation starts. With A30, only ASCII encoding is supported. SDMSIG shall always be mirrored within the file.

6.3.8.11 Signature Calculation

The key applied for the signature calculation is the ECCPrivateKey defined by FileAR.SDMFileRead2.

The SDMSIG is calculated using ECDSA Digital Signature Generation as defined in [12]. The hash function to be applied is SHA-256, as specified in NIST FIPS 180-4[17].

SDMSIG is defined as follows:

$SDMSIG = ECDSA_{Sign}(Priv.x,\ DynamicFileData[SDMMACInputOffset..SDMMACOffset - 1])$

with *DynamicFileData* being the file data as how it is put on the external interface, i.e. replacing any placeholders by the dynamic data.

6.3.8.12 SDM Session Key Generation

For Secure Dynamic Messaging for reading, the following session keys are calculated:

- SesSDMFileReadMACKey for MACing of file data.
- SesSDMFileReadENCKey for encryption of file data

The session key generation is according to NIST SP 800-108 [9] in counter mode.

The pseudo-random function applied during the key generation is the CMAC algorithm described in NIST Special Publication 800-38b [7]. The key derivation key is the SDMFileReadKey as configured with the SDMFileRead access right.

6.3.8.12.1 AES mode session key generation for SDM

The input data is constructed using the following fields as defined by [9]. NIST SP 800-108 allows defining a different order than proposed by the standard as long as it is unambiguously defined.

- A 2-byte label, distinguishing the purpose of the key: 0x3CC3 for MACing and 0xC33C for encryption.
- A 2-byte counter
  - KeyType.AES128: fixed to 0x0001.
  - KeyType.AES256: counting from 0x0001 to 0x0002.
- A 2-byte length,
  - KeyType.AES128: fixed to 0x0080.
  - KeyType.AES256: fixed to 0x0100.
- A context, constructed using the UID and/or SDMReadCtr, followed by zero-byte padding if needed.

Whether or not the UID and/or SDMReadCtr are included in session vector SV2, depends on whether they are mirrored, see Section 6.3.8.3. In case of encrypting file data, mirroring of both is mandatory.

Therefore, they are always included in SVx.

KeyType.AES128

First, the input session vectors SVx are derived as follows:

SV1 = 0xC3 || 0x3C || 0x00 || 0x01 || 0x00 || 0x80 || UID || SDMReadCtr

SV2 = 0x3C || 0xC3 || 0x00 || 0x01 || 0x00 || 0x80 [|| UID] [|| SDMReadCtr] [|| ZeroPadding]

Padding with zeros is done up to a multiple of 16 bytes. So, in case of 7-byte UID and both elements are mirrored, no padding is added. Then, the 16-byte session keys are constructed as follows:

SesSDMFileReadENCKey = MAC(SDMFileReadKey; SV1)

SesSDMFileReadMACKey = MAC(SDMFileReadKey; SV2)

KeyType.AES256

First, the input session vectors SVxy are derived as follows:

SV 1a = 0xC3||0x3C||0x00||0x01||0x01||0x00||V CUID||SDMReadCtr[||ZeroPadding]

SV 1b = 0xC3||0x3C||0x00||0x02||0x01||0x00||V CUID||SDMReadCtr[||ZeroPadding]

SV 2a = 0x3C||0xC3||0x00||0x01||0x01||0x00[||V CUID][||SDMReadCtr][||ZeroPadding]

SV 2b = 0x3C||0xC3||0x00||0x02||0x01||0x00[||V CUID][||SDMReadCtr][||ZeroPadding]

Padding with zeros is done up to a multiple of 16 bytes. So in the case of 7-byte UID and both elements are mirrored, no padding is added. Then, the 32-byte session keys are constructed as follows:

SesSDMFileReadENCKey = MAC(SDMFileReadKey; SV1a) || MAC(SDMFileReadKey; SV1b)

SesSDMFileReadMACKey = MAC(SDMFileReadKey; SV2a) || MAC(SDMFileReadKey; SV2b)

6.3.8.13 Output Mapping Examples

The following figure shows an example with the static file content and how it will be read.

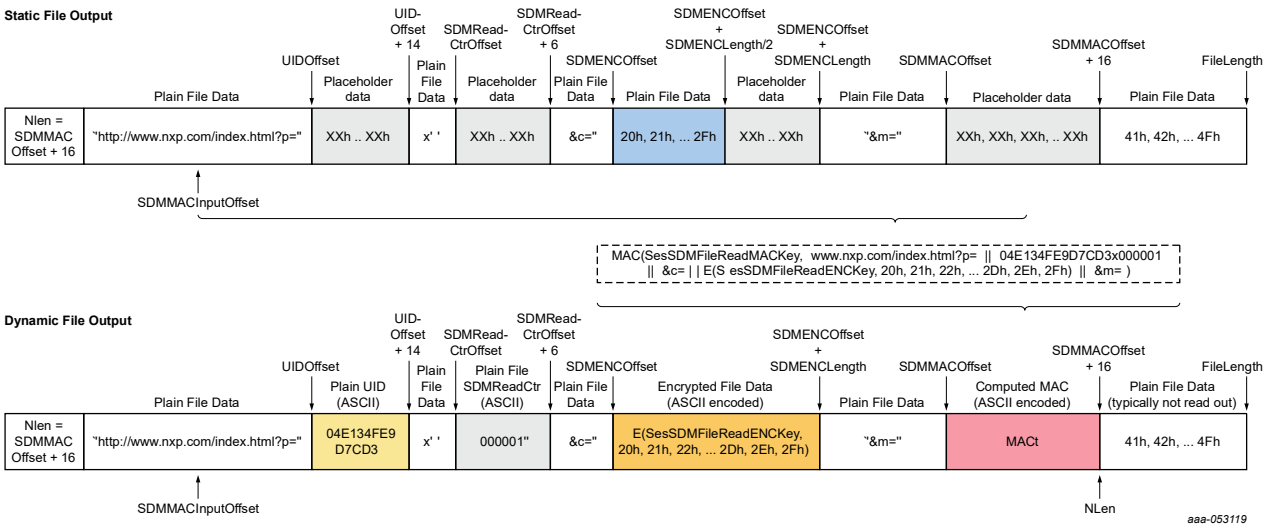


Figure 11. Secure Dynamic Messaging for Reading example

6.4 Access Rights Management

A30 manages its access rights through access conditions. This is explained in Section 6.4.1. How access rights can be granted through certificates presented during asymmetric authentication is explained in Section 6.4.3.

6.4.1 Access conditions

For file access, the conditions for the file access rights are associated with the file, as explained in Section 6.10.2. For other commands, the access conditions are either fixed or configurable via other means.

Nevertheless, the interpretation of access conditions and their representation in the command API is always the same. There are three kinds of access conditions:

- The *authentication* access conditions where a valid authentication is required. The access condition is satisfied by one of the following means:
  - an active symmetric authentication with the AuthKey addressed by the key number encoded by the access condition.
  - an active asymmetric authentication granting the access condition via the current CertAccessRights. This means the CARootKey addressed during the authentication must have been associated with access rights encoded by the access condition. How a CARootKey is configured with its access rights is defined in Section 6.4.2. Optionally the reader certificate (or certificate chain) presented during the authentication can further restrict the granted access rights from the CARootKey. This is specified in Section 6.4.3.
- The *free access over I^2C* condition meaning the related commands can be accessed without an active authentication over the I^2C interface.
- The *free access condition* meaning the related commands can be accessed without an active authentication over any interface.
- The *no access condition* meaning no access to the related commands.

Note: *In other parts of the document, when it is stated that an active authentication with AppKey is required, this means either a symmetric authentication with that particular key, or an asymmetric authentication granting equivalent access rights (even if the latter is not explicitly mentioned).*

The access conditions are specified on 4 bits as defined in Table 17.

Table 17. Access condition values coded on 4 bits

Condition value	Description
0x0..0xB	authentication required
0xC	authentication required over I^2C
0xD	free access over I^2C
0xE	free access
0xF	no access or RFU

A 0xC or 0xD access condition can also still be obtained over the respectively I^2C, if obtaining the access right from an ECC-based authentication.

Concretely, this means that the access conditions 0x0..0x4 can be obtained both through symmetric and asymmetric authentication, while the access conditions 0x5..0xD can be obtained through asymmetric authentication independently of the interface.

This is also illustrated by Figure 12 where access condition 0x0 can be obtained via a symmetric authentication targeting AppKey 0x00, i.e. the AppMasterKey, but also through an asymmetric authentication targeting CARootKey.1, with an ACMap set to 0x0021. This means that the latter also grants AppMasterKey access rights.

In the case of asymmetric authentication, the access rights granted depend on the targeted CARootKey, but can be further restricted via the certificate. If a certificate does not hold explicit access rights, the access rights from the related CARootKey are implicitly inherited and therefore granted. When authenticating CARootKey.1 of the example below, by default the accumulated access rights equivalent to being authenticated with either symmetric key 0x0 or key 0x5 will be granted. However, Cert.ReaderB handed out by the CA related with CARootKey.1, only grants access right 0x5, and therefore in that case not the AppMasterKey access rights. CARootKey.1 only has two bits set, but there is no limitation on the number of bits and therefore access rights that can be granted to a CARootKey: it could have all 14 bits set, granting access rights equivalent to the symmetric key 0x0 until 0xD.

Another example: If successfully authenticated with Cert.ReaderC, the user is granted access right 0x6 associated with the CARootKey.2, having its ACMap set to 0x0040. A CA shall not hand out certificates with access rights that exceed the access rights associated with the CARootKey. For example, a certificate with access right 0x3 handed out by the CA associated with CARootKey.1 will not be accepted by A30 as this access right is not configured for CARootKey.1.

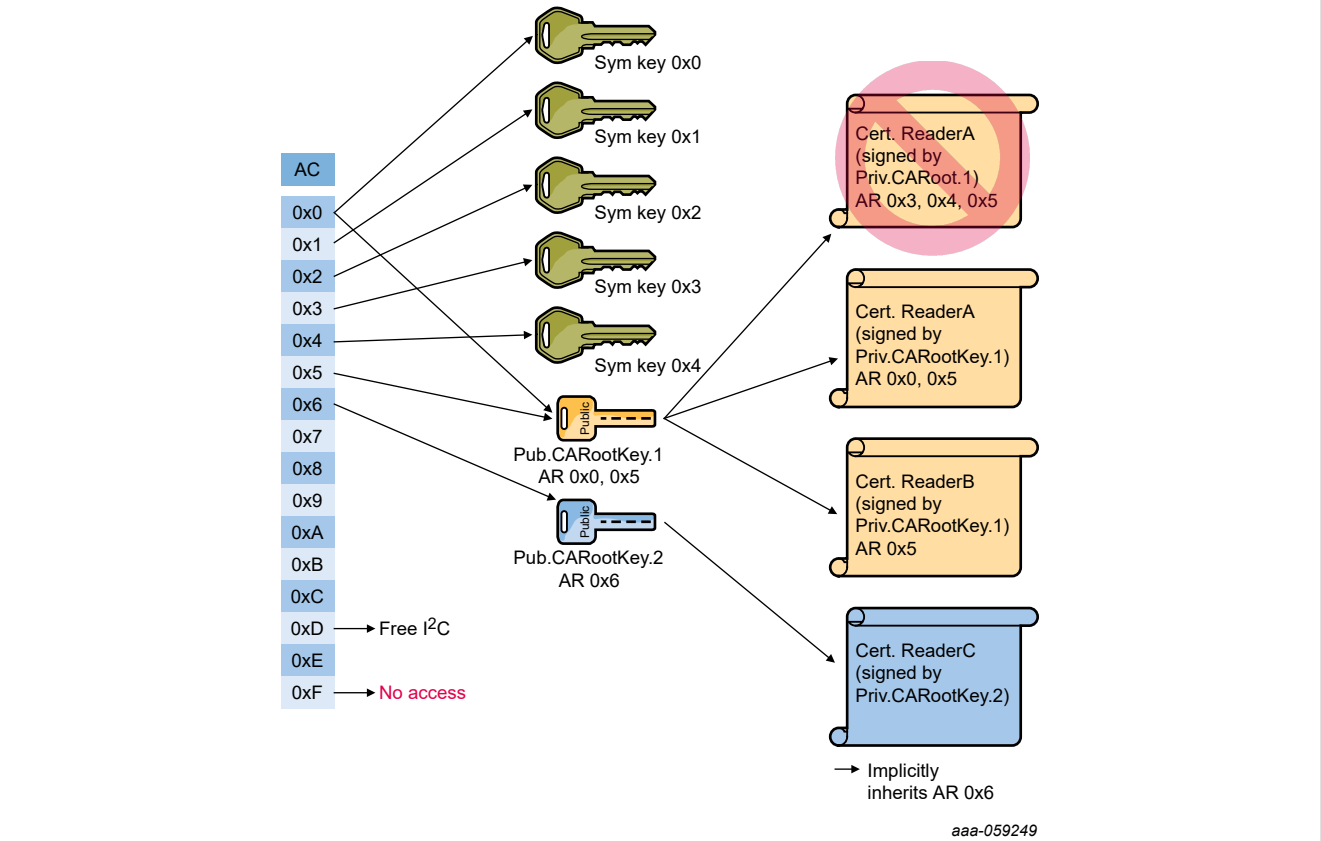


Figure 12. Access conditions example

6.4.2 CARootKey access rights

Access rights are associated with a CARootKey through ManageCARootKey. For this command, the access conditions that can be granted when authenticated with this CARootKey are encoded on a bitmap, as defined in Table 18. As defined in Section 6.15.1.1, this bitmap is sent on the interface LSB first.

Table 18. ACMap encoding

BitIndex	Description
Bit 15-14	RFU
Bit 13-0	AC bitmap. If bit 0 is set, AC 0x0 access rights are granted. If bit1 is set, AC 0x1 access are rights granted. And so on.

For these access rights the whole range of the AC bitmap can be used, independently of whether those bits encode key numbers of keys that exist in the targeted application. For example, if the application holds five symmetric keys, the key numbers 0x0-0x4 (i.e. bit 0 until bit 4 in the certificate encoding) can be used to specify access conditions that can be obtained by both symmetric and asymmetric authentication. From bit 5 onwards, the bits can only be used to specify access conditions that can be obtained via an asymmetric authentication, as there does not exist an equivalent symmetric key within the application.

Access rights obtained during a mutual authentication can be further restricted via the presented certificates, as defined in Section 6.4.3. There the same encoding is used.

6.4.3 Certificate access rights

A30 supports a private extension (ARExtension) for access right encoding within X.509 certificates. This extension is reflected by an OID in the NXP range: 1.3.6.1.4.1.28137.64.1. If not present, the CertAccessRights are inherited from the parent certificate, or in the case of no parent, the targeted CARootKey.

This access right extension will be processed independently of whether the criticality flag is set or not. A30 does not recognize any other extension. If a reader certificate with another extension marked critical is presented, it is rejected. If the criticality flag of an unrecognized extension is not set, the extension is ignored without rejecting the certificate.

This ARExtension has the following ASN.1 encoding:

```
ARExtension  - SEQUENCE {
          arExtnId OBJECT IDENTIFIER (id-nxp-ar),
          critical BOOLEAN (TRUE),
          arExtnValue OCTET STRING
}
```

CertAccessRights are obtained from a successful SIGMA-I authentication, see Section 6.3.2. They are maintained as long as in VCState.AuthenticatedECC.

The access right extension value (ARExtensionValue) shall hold the data structure as defined in Table 19. The actual length and value format depend on the ARType. The total length is also encoded in the OCTET STRING encoding of the extension.

Table 19. Application access rights, specified via DFName

Field	Length/Bit Index	Description
ARType	1	Tag specifying the type of ARG
	Bit 7	CA delegation
		'0': disabled (leaf)
		'1': enabled (parent or leaf)
	Bit 6-0	AR Type
		0x02: Application access rights, specified via DFName
ARValue	Variable	Actual access rights

Bit 7 of the ARType indicates whether the certificate can be used as a parent certificate delegating access rights. Only if the bit is set, the certificate can be used to compose a certificate chain, representing an intermediate CA. In this case the certificate can also still be used directly as a leaf certificate. If the bit is not set, the certificate can only be used as a leaf certificate. If used as a parent with the bit not set, the certificate validation fails.

In the case of application access rights, specified via DFName, as defined in Table 20, the ARValue consist of a variable length DFName, followed by a 2-byte ACMap. The latter defines the actual access rights granted for that application, as further defined in Table 18.

Table 20. Application access rights, specified via DFName

Field	Length	Value	Description
ARType	1	0x02/0x82	Access rights for application with the given DFName
DFNameLen	1	0x01.. 0x10	Length of the subsequent DFName of the application. This shall be set to 0x07 for A30.
DFName	DFNameLen	Full Range	DF Name of the application
ACMap	2	see Table 18	2-byte map of granted access conditions of the application

6.5 Card Memory and Configuration Management

6.5.1 Card Version

A30 is characterized by manufacturer-related data. These data are composed from HW-related information, SW-related information and production-related information as detailed in Table 21. For concrete response values, see Table 93.

Table 21. Manufacturer characteristics used as card version

Manufacturer characteristics	Size in bytes	Details
Hardware-related information		
Vendor ID	1	Identification of the card vendor, 0x04 for NXP Semiconductors.
HW type	1	Hardware platform type.
HW subtype	1	Hardware platform subtype.
HW major version number	1	Hardware platform major version number.
HW minor version number	1	Hardware platform minor version number.
HW storage size	1	Hardware platform storage size. See Table 93 for actual values.
HW protocol	1	Hardware communication protocol type.
Software-related information		
Vendor ID	1	Card vendor identification.
SW type	1	Card software type.
SW subtype	1	Card software subtype.
SW major version	1	Card Software major version number: reflects the evolution(EVx) and is only incremented on major feature introduction.
SW minor version	1	Card Software minor version number: consists of SW minor (upper nibble) and sub-minor (lower nibble) version. SW minor version will be incremented if introducing new features or feature extensions not justifying an SW major increment. SW sub minor will be incremented on patched versions or very minor feature extensions.
SW storage size	1	Card Software storage size. See Table 97 for actual values.
SW protocol	1	Card Software communication protocol type.

Table 21. Manufacturer characteristics used as card version*...continued*

Production-related information		
UID	7	Card unique identifier as defined in Card UID. If the Random ID is activated always 7 0x00 bytes are returned. When switching to random ID this is only reflected after reset and reactivation.
Batch number	3	Fabkey server batch number.
FabKeyID	2	Fabkey identifier in alphanumeric ASCII encoding
CW production	1	Calendar week of card production in BCD coding (i.e. week36 is code as 0x36).
Year of production	1	Year of card production in BCD coding (i.e. year 2012 is code as 0x12).
Fab ID	[1]	Fab Identifier, only present if requested via Option byte.

For enhanced privacy, A30 supports an option to mask the manufacturer data, i.e. Batch Number, CW production, Year of production and FabKey (as a consequence FabKeyID will not be present). This masking of manufacturer data can be configured with SetConfiguration Option 0x0E, see Section 6.5.1. If enabled, this is independent of the Random ID configuration, and of whether there is an active authentication.

6.5.1.1 Command GetVersion

Reading the version of a card as defined in Section 6.5.1 is possible with the command GetVersion as defined in Section 7.4.5.

No parameters are passed with this command.

The version data is return over three frames. As specified in Table 21, 1st Frame returns the hardware-related information, 2nd returns the software-related information, and the 3rd and last frame returns the production-related information.

This command does not require authentication. If there is an active authentication, the command GetVersion requires CommMode.MAC. Information on the authentication and secure messaging-dependent structure of the command can be found in Section 6.3.

6.5.2 Card configuration

6.5.2.1 Command SetConfiguration

SetConfiguration is updating configuration settings. Its specifications can be found in SetConfiguration. The command consists of an option byte and a data field with a size depending on the option.

In the below table, “No change” references are used with configurations that are persistent. This means that the associated configuration is left as it is already in the card and its value is not changed.

Table 22. SetConfiguration options list

Option byte	Field	Length/ BitIndex	Description
0x00 .. 0x03			Reserved
0x04		**Total: 2**	**SecureMessaging Configuration**
	SMConfigA	1	Secure messaging configuration (Byte A)
		Bit 7-3	RFU
		Bit 2	EV2 secure messaging configuration for FileType.StandardData
			0:No change
			1: In VCState.AuthenticatedAES and VCState.AuthenticatedECC, disable chained writing with WriteData in CommMode.MAC and CommMode.Full.
		Bit1-0	Reserved
	SMConfigB	1	Secure messaging configuration (Byte B)
		Bit 7-0	RFU
0x05 .. 0x0F			Reserved
0x10		**Total: 4**	**I^2C Management**
	I2CSupport	1	I^2C Support
		Bit 7-1	RFU
		Bit 0	I^2C I/O
			0: I^2C disabled
			1: I^2C enabled (default)
	I2CAddress	1	The address used for the I^2C target (default 0x20)
	ProtocolOptions	2	The crypto protocols supported over I^2C. See Table 23. The default value is that all protocols supported in the manufacturing features selection map are enabled and Protocol Negotiations disabled.
0x11		**Total: 28**	**GPIO Management**
	GPIO1Mode	1	GPIO1 Mode
			0x00: disabled (default)
			0x01: input
			0x02: output
			0x03: input tag tamper
			0x04: down-stream power out
	GPIO1Config	1	GPIO1 Configuration, see Table 24.
	GPIO1PadCtrl	4	GPIO1 Pad Control, see Table 25.
	GPIO2Mode	1	GPIO2 Configuration
			0x00: disabled (default)
			0x01: input
			0x02: output

Table 22. SetConfiguration options list...*continued*

	GPIO2Config	1	GPIO2 Configuration, see Table 24.
	GPIO2PadCtrl	4	GPIO2 Pad Control, see Table 25.
	GPIO1Notif	1	GPIO notification on authentication. Note: ***Note:*** *Notification is only allowed if GPIO1Mode is 0x02.*
			0x00: disabled (default)
			0x01: enable authentication notification
	GPIO2Notif	1	GPIO notification on authentication. ***Note:*** *Notification is only allowed if GPIO2Mode is 0x02.*
			0x00: disabled (default)
			0x01: enable authentication notification
	ManageGPIO-AccessCondition	1	ManageGPIO access condition
		Bit 7-6	RFU
		Bit 5-4	CommMode, see Table 14.
		Bit 3-0	AccessCondition Value, see Table 17. Default 0xF.
	ReadGPIO-AccessCondition	1	ReadGPIO access condition
		Bit 7-6	RFU
		Bit 5-4	CommMode, see Table 14.
		Bit 3-0	AccessCondition Value, see Table 17. Default 0xF.
	DefaultTarget	1	[Applicable when GPIO1Mode = 0x04] Targeted voltage/current level
			0x00: disable in rush current limit.
			0x01: power downstream of 1.8 V and 100 µA
			0x02: power downstream of 1.8 V and 300 µA
			0x03: power downstream of 1.8 V and 500 µA
			0x04: power downstream of 1.8 V and 1 mA
			0x05: power downstream of 1.8 V and 2 mA
			0x06: power downstream of 1.8 V and 3 mA
			0x07: power downstream of 1.8 V and 5 mA
			0x08: power downstream of 1.8 V and 7 mA
			0x09: power downstream of 1.8 V and 10 mA
			0x11: power downstream of 2 V and 100 µA
			0x12: power downstream of 2 V and 300 µA
			0x13: power downstream of 2 V and 500 µA
			0x14: power downstream of 2 V and 1 mA
			0x15: power downstream of 2 V and 2 mA
			0x16: power downstream of 2 V and 3 mA
			0x17: power downstream of 2 V and 5 mA
			0x18: power downstream of 2 V and 7 mA

Table 22. SetConfiguration options list...*continued*

			0x19: power downstream of 2 V and 10 mA
			0x1F: power downstream of 2 V and MAX available current
	InRushTarget	1	[Applicable when GPIO1Mode = 0x04] Initial current limit to handle the in rush of current when charging an external capacitor.
			0x00: disable in rush current limit.
			0x01: power downstream of 1.8 V and 100 µA
			0x02: power downstream of 1.8 V and 300 µA
			0x03: power downstream of 1.8 V and 500 µA
			0x04: power downstream of 1.8 V and 1 mA
			0x05: power downstream of 1.8 V and 2 mA
			0x06: power downstream of 1.8 V and 3 mA
			0x07: power downstream of 1.8 V and 5 mA
			0x08: power downstream of 1.8 V and 7 mA
			0x09: power downstream of 1.8 V and 10 mA
			0x11: power downstream of 2 V and 100 µA
			0x12: power downstream of 2 V and 300 µA
			0x13: power downstream of 2 V and 500 µA
			0x14: power downstream of 2 V and 1 mA
			0x15: power downstream of 2 V and 2 mA
			0x16: power downstream of 2 V and 3 mA
			0x17: power downstream of 2 V and 5 mA
			0x18: power downstream of 2 V and 7 mA
			0x19: power downstream of 2 V and 10 mA
			0x1F: power donwstream of 2V and MAX available current
	InRushDuration	2	[Applicable when GPIO1Mode = 0x04] The duration to ap- plythe InRushTarget
			0x0000.. 0xFFFF: targeted duration in ms.
	AdditionalCurrent	1	[Applicable when GPIO1Mode = 0x04] The additional current required by A30 when supplying power harvesting. Resolution: 0.4 mA.
			0x00..0x1F
0x12		**Total: 2**	**ECCKey Management**
	ManageKeyPair AccessCondition	1	ManageKeyPair access condition
		Bit 7-6	RFU
		Bit 5-4	CommMode, see Table 14. Default '11'.
		Bit 3-0	AccessCondition Value, see Table 17. Default 0x0.
	ManageCARoot KeyAccess Condition	1	ManageCARootKey access condition

Table 22. SetConfiguration options list...*continued*

		Bit 7-6	RFU
		Bit 5-4	CommMode, see Table 14. Default '11'.
		Bit 3-0	AccessCondition Value, see Table 17. Default 0x0.
0x13		**Total: 4**	**CertificateManagement**, see Section 6.8
	LeafCacheSize	1	End Leaf certificate cache size: number of slots in end leaf cert cache. The certificate cache is created the first time that it is enabled or on the first boot if enabled by default. The cache sizes are ignored after the certificate cache has been created.
			0x01..0x08
	IntermCacheSize	1	Intermediate certificate cache size: number of slots in end leaf cert cache. The certificate cache is created the first time that it is enabled or on the first boot if enabled by default. The cache sizes are ignored after the certificate cache has been created.
			0x02..0x08
	FeatureSelection	1	Feature Selection
		Bit 7-5	RFU
		Bit 4-3	HostCertificate Support
			01: support full host certificates
			Other: RFU
		Bit 2-1	InternalCertificate Support
			00: repository default (use certs as stored in the certificate repo)
		Bit 0	EnableSIGMA-I cache
			0: disabled (default)
			1: enabled
	ManageCert Repo- Access Condition	1	ManageCertRepoCreate Option ('00') Access Condition
		Bit 7-6	RFU
		Bit 5-4	CommMode, see Table 14. Default '10'.
		Bit 3-0	AccessCondition Value, see Table 17. Default 0x0. The read and write/reset certificate repository access conditions are set during repository creation.
0x14		**Total: 3**	**WatchdogTimer Management**, see Section 6.14
	HWDTValue	1	HaltWatchdog Timer (HWDT) Value. Maximum time before A30 shall enter the hardware HALT state. The timer is started by device reset or when the device exits the Halt state. It is reset by command reception on I2C.
			0x00: disabled (default)
			0x01..0x3C: 1 s -60 s
	AWDT1Value	1	Authorization watchdog Timer (AWDT1) Value. The maximum time before the A30 shall abort an authentication attempt, be it via SIGMA-I or symmetric mutual authentication.
			0x00: disabled (default)
			0x01..0x3C: 1 s-60 s

Table 22. SetConfiguration options list...*continued*

	AWDT2Value	1	Authorization watchdog Timer (AWDT2) Value. The maximum time beforeA30 shall revoke current authentication status, be it from SIGMA-I or symmetric mutual authentication.
			0x00: disabled (default)
			0x01..0x3C: 1 s-60 s
0x15		**Total: 5+M3+N*3**	**CryptoAPI Management**
	Support	1	CryptoAPI Support
		Bit 7-2	RFU
		Bit 1	AsymmetricCrypto API
			'0': disabled
			'1': enabled (default)
		Bit 0	SymmetricCrypto API
			'0': disabled
			'1': enabled (default)
	AccessCondition	1	Access condition for CryptoRequest
		Bit 7-6	RFU
		Bit 5-4	CommMode, see Table 14. Default CommMode.MAC.
		Bit 3-0	AccessCondition Value, see Table 17. Default 0x0.
	ChangeAC	1	Access condition for ChangeKey targeting CryptoRequestKey.
		Bit 7-4	RFU
		Bit 3-0	AccessCondition Value, see Table 17. Default 0x0.
	TBPolicyCount	1	CryptoAPI Transient Buffer Policy Count (M)
			0x00..0x08
	TBPolicy	M*3	Crypto API Transient Buffer Policy, see Section 6.12. Each 3-byte instance consists of: Destination (see Table 38) || Usage Policy (see Table 39) || Algorithm Policy (see Table 40)
	SBPolicyCount	1	CryptoAPI Static Buffer Policy Count (N)
			0x00..0x0E
	SBPolicy	N*3	CryptoAPI Static Buffer Policy, see Section 6.12. Each 3-byte instance consists of: Destination (see Table 38) || Usage Policy (see Table 39) || Algorithm Policy (see Table 40)
0x16		**Total: 6**	**AuthenticationCounter and Limit Configuration**
	AuthCtrFileID	1	Targeted FileType.Counter
			0x00..0x1F: FileID of the targeted file
	AuthCtrOption	1	Authentication counter options
		Bit 7-1	RFU
		Bit 0	AuthenticateEV2First (AES-based authentication)

Table 22. SetConfiguration options list...*continued*

			’0’: disabled (default)
			’1’: enabled
	AuthCtrLimit	4	Authentication Counter Limit
			0x00000000:AuthCtrLimit disabled
			0x00000001.. 0xFFFFFFFF:AuthCtrLimit enabled with the given value
0x17		**Total: 4**	**HALTandWake-upConfiguration**
	WakeUpA	1	Wake-up options (Byte A)
		Bit 7	RFU
		Bit 6	GPIOwake-up: GPIO2 pulldown triggers wake-up.
			’0’: disabled (default)
			’1’: enabled
		Bit 5-0	I^2C wake-up address (default: 0x20): 6 MSB of 7-bi I2C address used for wake-up
			Full range
	WakeUpB	1	Wake-up options (Byte B)
		Bit 7	I^2C wake-up address (default: 0x20): LSB of 7-bit I2C wake-up Address
			Full range
		Bit 6-3	I2CSDA wake-up cycles (default: 0x0): number of SCL cycles that are required to wake up when SDA is pulled down.
			Full range
		Bit 2	I^2C address wake-up: If targeted I2C address matches the configured I2C wake-up address, wake-up is triggered.
			’0’: disabled
			’1’: enabled (default)
		Bit 1	I2C SDA cycle wake-up: If I2C SDA cycles match the configuredI2C wake-up cycles, wake-up are triggered.
			’0’: disabled (default)
			’1’: enabled
	RDACSetting	1	RDACSetting: impacts how much energy is drawn from theRF field, while the device is in HALT state.
			0x00..0xFF (default: 0x00)
	HALTOption	1	HALT options
		Bit 7-2	RFU
		Bit 1	GPIO2 reset: before entering power-saving HALT state, GPIO2 pin resets to High-Z state.
			’0’: disabled
			’1’: enabled (default)
		Bit 0	GPIO1 reset: before entering a power-saving HALT state, GPIO1 pin resets to High-Z state.

Table 22. SetConfiguration options list...*continued*

			'0': disabled
			'1': enabled (default)
0x18 .. 0xFD			RFU
0xFE		**Total: 1+N*2**	**Defer Configurations**
	DeferralCount	1	DeferralCount (N)
			0x01..0x03
	DeferralList	N*2	List of Deferrals. See Table 1.
0xFF		**Total: 3**	**Lock Configurations**
	LockMap	3	Bitmap where each bit encodes for the related configuration option if it is locked. LSB first, i.e. first byte encodes Option 0x07-0x00.
		Bit 23-0	Lock bit
			'0': No Change
			'1': Lock configuration

Table 23. ProtocolOptions

Field	Length/ BitIndex	Description
ProtocolOptionsA	1	ProtocolOptions (Byte A)
	Bit 7	Controller session key usage, see Section 6.3.7
		'0': Disabled (default)
		'1': Enabled
	Bit 6-4	RFU
	Bit 3	Reserved
	Bit 2	ECC-based Card-Unilateral authentication (ISOInternalAuthenticate)
		'0': Disabled
		'1': Enabled (default)
	Bit 1	Reserved
	Bit 0	AES-based Symmetric authentication (AuthenticateEV2First, AuthenticateEV2NonFirst)
		'0': Disabled
		'1': Enabled (default)

Table 23. ProtocolOptions...*continued*

ProtocolOptionsB	1	ProtocolOptions (Byte B)
	Bit 7-5	RFU
	Bit 4	Enable SIGMA-I Verifier for host(ISOGeneralAuthenticate with P1=0x01where host acts as initiator, i.e. starts with 0xA0 message)
		'0': Disabled
		'1': Enabled (default)
	Bit 3	Enable SIGMA-I Prover for host (ISOGeneralAuthenticate with P1=0x01where host acts as responder, i.e. starts with 0xB0 message)
		'0': Disabled
		'1': Enabled (default)
	Bit 2	Secure Tunnel variant after SIGMA-I authentication (ISOGeneralAuthenticate with P1=0x01)
		'0': NTAG EV2 secure messaging
	Bit 1	Secure Tunnel strength for SIGMA-I authentication (ISOGeneralAuthenticate with P1=0x01)
		'0': AES-256 not supported
		'1': AES-256 supported (default)
	Bit 0	Secure Tunnel strength for SIGMA-I authentication (ISOGeneralAuthenticate with P1=0x01)
		'0': AES-128 not supported
		'1': AES-128 supported (default)

Table 24. GPIOxConfig

Field	Length/ BitIndex	Description
GPIOxConfig	1	GPIOx Configuration
	-	**[ifGPIOxMode is output (0x02 or 0x05)]**
	Bit 7-1	RFU
	Bit 0	Initial state after power-off cycle
		0:Low (i.e. equivalent to after CLEAR operation with ManageGPIO)
		1:High (i.e. equivalent to after SET operation with ManageGPIO)
	-	**[elseif GPIOxMode is down-stream power out (0x04)]**
	Bit 7-2	RFU
	Bit 1	I2C Support
		0: disabled
		1: enabled
	-	**[else]**
	Bit 7-0	RFU

Table 25. GPIOxPadCtrl

Field	Length/ BitIndex	Description
GPIOxPadCtrlA	1	GPIOx Pad Control (Byte A)
	Bit7-2	RFU
	Bit1-0	DebounceFilter value: the 2 MS bits of the 10 bit debounce filter value.
GPIOxPadCtrlB	1	GPIOx Pad Control (Byte B)
	Bit7-0	Debounce Filter value (Resolution = 0.1 µs): the LS 8 bits of the 10 bit debounce filter value. Bit 0 is the LSB. Writing a value of 1 filters out glitches less than 0.1 µs. Writing a value of 1000 (over the10 bits) filters out glitches less than 100us
GPIOxPadCtrlC	1	GPIOx Pad Control (Byte C)
	Bit7-3	RFU
	Bit 2	Debounce filter
		0:Disable debounce filter
		1:Enable debounce filter of min.5us/max.60us
	Bit1-0	Input filter selection
		00: unfiltered input selected, (filter of 50 ns selected but has no effect)
		01: unfiltered input selected, (filter of 10 ns selected but has no effect)
		10: ZIF filtered input selected, filter of 50 ns selected
		11: ZIF filtered input selected, filter of 10 ns selected

Table 25. GPIOxPadCtrl...*continued*

GPIOxPadCtrlD	1	GPIOx Pad Control (Byte D)
	Bit7-5	Input configuration
		000: Plain input with weak pullup
		001: Plain input with repeater (bus keeper)
		010: Plain input
		011: Plain input with weak pulldown
		100: Weak pullup
		101: Weak pulldown (*DISABLE_WPDN*)
		110: High-impedance (analog I/O)
		111: Weak pulldown (*DISABLE_WPD*)
	Bit4-1	Output configuration
		0000: I^2C S/F and FP Transmit mode (SDA and SCL) and I^2C HStransmit mode (only S0xDA)
		0001: I^2C HS Transmit mode (only SCLK)
		0010: *I^2C_TX_SFFP*
		0011: *I^2C_TX_HS_SCLK*
		0100: GPIO Low-speed mode (*GPIO_LOW_SPEED*_1)
		0101: GPIO Low-speed mode (*GPIO_LOW_SPEED*_2)
		0110: GPIO High-speed mode (*GPIO_HIGH_SPEED*_1)
		0111: GPIO High-speed mode (*GPIO_HIGH_SPEED*_2)
		1000-1111: Output disabled
	Bit 0	Supply selection
		0: 1V8 signaling in I^2C mode
		1: 1V1 and 1V2 signaling in I^2C mode

Each of the supported options of the command SetConfiguration requires active authentication granting AppMasterKey access rights.

The command SetConfiguration requires CommMode.Full. Information on the authentication and secure messaging-dependent structure of the command can be found in Section 6.3.

The command is rejected if:

- required authentication is not active.

Options 0x0F and 0x10 configure the communication interfaces of the product, and what cryptographic protocols are available over each interface. Extreme care must be taken when configuring these options as e.g.disabling both interfaces makes the product unusable. Also, option 0x10 does not check the provided I2CAddress against reserved addresses as specified in [15].

Option 0x11 allows for configuring the GPIO pins and related access conditions for further managing and/or reading them, see also Section 6.13. Values related to specific modes are not checked for consistency. This means it is the user responsibility to provide a meaningful configuration. Even if not applicable for the configured mode, the provided values are stored and returned by GetConfiguration.

A change in the GPIO configuration is guaranteed after the next power-off reset.

Option 0x14 allows timers as detailed in Section 6.14. A change in the HWDT configuration is guaranteed after the next power-off reset.

Option 0x15 configures some aspects of the generic Crypto API, see Section 6.12. Asymmetric and symmetric cryptography can be separately enabled through the Support byte. AccessCondition defines the communication mode and required access rights for CryptoRequest. The default is CommMode.MAC and AppMasterKey access rights. ChangeAC defines the access condition for changing CryptoRequestKeys with ChangeKey. The default is requiring AppMasterKey access rights. Next to these slot policies for the Transient and Static Buffers slots can be configured. By default, those are set to 0x0000, i.e. they must be configured to enable Transient and Static Buffer usage. It is recommended to configure more strict policies depending on the targeted use case, especially if the CryptoRequest is changed to free or free over I^2C access. Any policy that is not explicitly updated remains unchanged.

Option 0xFF allows deferring some configurations, see Deferred Configuration Options.

Option 0xFF allows locking the other configurations. A bitmap *Lock map* is to be provided. This is sent LSB first, i.e. to lock Option 0x00, Bit 0 of the first transmitted byte must be set. Setting a bit for a nonsupported option (RFU or Reserved) does not have any effect. Once a configuration is locked, it cannot be unlocked, i.e. setting a bit to 0 does not change the current state.

All configurations can on request get customer-specific values through the OEF specification, instead of the default values listed here.

6.5.2.2 Command GetConfiguration

GetConfiguration is retrieving card or application configuration settings. Its specifications can be found in Section 7.4.3.

If no configuration option byte is specified, then manufacturer data like the NXP Product Features Map is returned. When retrieving the Crypto API Management, i.e. Option 0x15, always all eight TB Policies are returned. If a policy has not been configured explicitly, the default value of 0xFFFF is returned.

The GetConfiguration is rejected at the PICC level.

The GetConfiguration is subject to the same access restrictions as the SetConfiguration i.e.it is subject to CommMode.Full, requiring AppMasterKey access rights.

6.5.2.3 Memory management

The nonvolatile memory available for user data is allocated in blocks of 32 bytes.

The user memory is available for creation of the following data items (including overhead):

- FileType.StandardData files and FileType.Counter files, see Section 6.7.5.

Table 26. Supported memory configurations

Memory configuration	Size in bytes	Size in blocks
8.5 kB	8704	0x0110
16.5 kB	16896	0x0210

Commands, which have an impact on the memory structure itself activate an automatic mechanism that protects the application and file structure from getting corrupted. If the card is unpowered during command execution, it is ensured that on the next activation the memory structure is automatically updated such that the card behaves either exactly as it was before the command execution, or as it would have been after having completed a successful execution.

6.5.2.3.1 Free Memory with Command FreeMem

The available free user memory on the card is returned with FreeMem as defined in Section 7.4.1.

No parameters are passed with this command.

The memory size in bytes available is returned as an unsigned integer.

If the PICC is authenticated, the command FreeMem requires CommMode.MAC. Otherwise, FreeMem is transmitted in plain. Information on the authentication and the secure messaging-dependent structure of the command can be found in Section 6.3.

6.6 Symmetric Key Management

6.6.1 Key Types

A30 supports symmetric key types as defined in Table 27.

As shown in the table, the different key types are represented by two bits.

Table 27. Supported key types

KeyType	BitRepresentation	Description
KeyType.AES128	10	AES-128keys
KeyType.AES256	11	AES-256keys

This representation is used at several places in the document.

KeyType.AES128 and KeyType.AES256 keys are stored in resp. 16 bytes or 32 bytes and are handled according to [5].

6.6.2 Key Versioning

A30 supports the versioning of symmetric keys by relating with each key a 1 byte key version number. The version of any addressable symmetric key can be read using GetKeyVersion.

6.6.3 Symmetric Keys

Symmetric application keys and their usage are defined in Table 28. They are used to manage the security of the application like file access control. Some of them can have additional roles assigned, like being required for key changing. An overview is given in Section 6.6.3.1.

These roles and other key related configurations are defined via the key settings, see Section 6.6.3.2.

Table 28. Keys at application level

Key Identifier	Key number	Change Key	Can be used for Authentication
Addressable keys:			
AppMasterKey	0x00	AppMasterKey	yes
AppKey	0x00..0x04	AppMasterKey	yes
SDMMetaReadKey	0x00..0x04	AppMasterKey	yes
SDMFileReadKey	0x00..0x04	AppMasterKey	yes
AppPrivacyKey	0x00..0x04	AppMasterKey	yes

Table 28. Keys at application level*...continued*

Key Identifier	Key number	Change Key	Can be used for Authentication
CryptoRequestKey	0x10..0x17	Configured via Set Configuration Option 0x15 ChangeAC	no

6.6.3.1 AppMasterKey

The AppMasterKey always has the key number 0x00.

The AppMasterKey can be a KeyType.AES128 or KeyType.AES256 key as set when changing the key with ChangeKey. When changing the key type of the AppMasterKey, the key type of all AppKeys change.

A successful authentication with the AppMasterKey is required to change any application key including the AppMasterKey itself with the ChangeKey command.

6.6.3.2 AppKey

The application of the A30 includes 5 application keys with key numbers 0, 1, 2, 3, 4.

At delivery, all AppKeys will be set to the default value of all zero bytes for key value and version, having KeyType.AES128, or they can be set via trust provisioning, see Section 6.16.3.

A AppKey can be a KeyType.AES128 or KeyType.AES256 key depending on the key type of the AppMasterKey.

The AppKeys are changeable with ChangeKey with an active authentication with AppMasterKey.

Remark: If not done through trust provisioning, it is highly recommended to change all 5 keys at personalization, even if not all keys are used in the application.

6.6.3.3 SDMMetaReadKey

The SDMMetaReadKey is one of the 5 AppKey. Which key is used is configured via ChangeFileSettings by adjusting the SDMMetaRead access rights, see Section 6.10.2.1. SDMMetaReadKey is used to encrypt PICCData before mirroring.

As the SDMMetaReadKey refers to an AppKey, it is changeable with ChangeKey with an active authentication with the AppMasterKey.

As the SDMMetaReadKey refers to an AppKey, it is available for authentication.

6.6.3.4 SDMFileReadKey

The SDMFileReadKey is one of the 5 AppKey. Which key is used is configured via ChangeFileSettings by adjusting the SDMFileRead access rights, see Section 6.10.2.1. SDMFileReadKey is used for Secure Dynamic Messaging.

As the SDMFileReadKey refers to an AppKey, it is changeable with ChangeKey with an active authentication with the AppMasterKey.

As the SDMFileReadKey refers to an AppKey, it is available for authentication.

6.6.3.5 AppPrivacyKey

The AppPrivacyKey is the AppKey identified by the key number specified with SetConfiguration Option 0x0E if this feature is enabled.

Once enabled, authentication with this AppPrivacyKey is required for GetCardUID.

6.6.3.6 CryptoRequestKey

The CryptoRequestKeys can be used for generic cryptographic operations.

An CryptoRequestKey is a KeyType.AES128 or KeyType.AES256 key.

The CryptoRequestKeys are changeable with ChangeKey according to the ChangeAC configuration of SetConfiguration Option 0x15. By default, an active authentication with AppMasterKey is required. When changing the key, the key type is defined, i.e. KeyType.AES128 or KeyType.AES256, and potential restrictions on the usage are specified through the given KeyPolicy. At delivery, by default, this KeyPolicy is set to 0x0000, i.e. disabling the key for any functionality.

The CryptoRequestKeys are not available for authentication.

6.6.4 Key Management Commands

This section gives the overall description of the key management commands as most of them apply to both PICC and application level.

6.6.4.1 Command ChangeKey

Changing keys is possible with the command ChangeKey as defined in Section 7.5.1. The command is also rejected if there is no active authentication with the relevant change key.

For the application level, all AppKeys, including AppMasterKey, require authentication with AppMasterKey. CryptoRequestKeys require authentication granting SetConfiguration Option 0x15 ChangeAC access rights. By default this is also AppMasterKey authentication. The required access rights can also be achieved through asymmetric authentication, see Section 6.4.

Under EV2 Secure Messaging, i.e. if in VCState.AuthenticatedAES or in VCState.AuthenticatedECC, the secure messaging is as applied under CommMode.Full, see Section 6.3.6.9. For the plaintext, two cases can be distinguished when targeting AppKeys, i.e. KeyNo 0x00 until 0x04:

Targeted key equal to authenticated key If the targeted key is equal to the authenticated key (i.e. *KeyNo*==*getKeyNo*(*AuthKey*), the plaintext is constructed as follows:

- KeyType.AES128: KeyData = *NewKey*||*KeyVer*(16 + 1 byte)
- KeyType.AES256: KeyData = *NewKey*||*KeyVer*(32 + 1 byte)

*NewKey*isthe new key. *KeyVer* is the related key version.

The normal EV2 secure messaging for CommMode.Full is applied on the command. The response is sent in plain, as the authentication is lost (see below).

In VCState.AuthenticatedECC, this case cannot occur. In VCState.AuthenticatedAES, this case applies only if targeting AppMasterKey.

Targeted key different from authenticated key If the targeted key is not equal to the authenticated key (i.e.

KeyNo!= *getKeyNo*(*AuthKey*)),the plaintext is constructed as follows:

- KeyType.AES128:KeyData = (*NewKey*⊕*OldKey*)||*KeyVer*||*CRC*32*NK*(16+ 1 + 4 byte)
- KeyType.AES256:KeyData = (*NewKey*⊕*OldKey*)||*KeyVer*||*CRC*32*NK*(32+ 1 + 4 byte)

NewKey is the new key value, while *OldKey* is the old key value currently present in the targeted key entry. If the key types of the *NewKey* and*OldKey* differ, the *OldKey* is truncated or padded with zeros to match the target key type size. *KeyV er* is the new version.

The *CRC*32*NK* is the 4-byte CRC value computed over *NewKey. The* CRC is computed according to IEEE Std 802.3-2008 (FCS Field)[20].

The normal EV2 secure messaging for CommMode.Full is applied on both the command and the response.

Note: *In VCState.AuthenticatedECC, this case always applies. In VCState.AuthenticatedAES, this case applies always if not targeting AppMasterKey.*

When targeting CryptoRequestKeys, i.e. KeyNo 0x10 until 0x17, always the first case applies. This means it is not required to proof knowledge of the old key for CryptoRequestKeys. The plaintext consists of the new key value concatenated with the key version, i.e. *NewKey||KeyVer*. Depending on the *ChangeAC* access condition, key updating of CryptoRequestKeys may be allowed in VCState.NotAuthenticated. In this case, the *KeyData* is sent in plain as no secure messaging applies. If used, it must be judged, via a system security assessment on the targeted use case, if this configuration creates a security risk.

The key value (*NewKey*) and the related key version (*KeyVer*) retrieved are used to change the targeted key. If the length does not match with the targeted key type, the command is rejected.

If targeting the AppMasterKey or CryptoRequestKeys, the key type is updated with the type specified in *KeyNo*[b7..6]. When changing the key type of the AppMasterKey, also the key type of all other AppKeys change, by truncating the key values if changing from KeyType.AES256 to KeyType.AES128, or padding with zero bytes if changing from KeyType.AES128 to KeyType.AES256.

If targeting CryptoRequestKeys, via the *KeyPolicy*, which is only present in this case, the allowed cryptographic functionality that can be executed with the targeted key can be restricted. It is not allowed to enable for a key both HMAC-based (bit 8-7) and AES-based (bit 6-0) algorithms.

If the key used for current active authentication AuthKey is changed, then the authentication is invalidated. The PICC moves into VCState.NotAuthenticated.

6.6.4.2 Command GetKeySettings

Retrieving key settings is possible with the command GetKeySettings as defined in Section 7.5.2.

At application level, an authentication with the AppMasterKey is required. At PICC level, where only option 0x01 is supported, no authentication is required.

If no Option is given, the following values are returned:

- *KeySetting* is set to 0x03, i.e. compatible to the AppKeySettings on a MIFARE DESFire product.
- Bit 7-6 of *MaxNoOfKeys* represents the key type of the application, encoded as defined in Table 27. This key type can be changed through updating the AppMasterKey. Bit5-0 is set to the number of application keys, i.e. 0x05.

If an Option is given, the metadata of a specific key group is returned.

Table 29. GetKeySettings Key Groups

Option	KeyGroup
0x00	CryptoRequestKeys
0x01	ECCPrivateKeys
0x02	CARootKeys

Under active authentication, the command GetKeySettings requires CommMode.MAC. Information on the authentication and the secure messaging-dependent structure of the command can be found in Section 6.3.

6.6.4.3 Command GetKeyVersion

Getting the key version of an addressable key is possible with the command GetKeyVersion as defined in Section 7.5.3.

KeyNo indicate which information is requested. If the key does not exist, the command is rejected. When retrieving a key version, a single byte *KeyVer* is returned holding the key version.

This command can be issued without an active authentication, but if there is an active authentication the command GetKeyVersion requires CommMode.MAC. Information on the authentication and the secure messaging-dependent structure of the command can be found in Section 6.3.

6.7 Asymmetric Key Management

A30 distinguishes private and public keys and the way that they are managed:

- ECCPrivateKey: This is the private key of an asymmetric ECC key pair, which is used to authenticate the A30 toward external parties. The management of these keys is detailed in Section 6.7.1.
- CARootKey: This is the public key of an asymmetric ECC key pair, which is used to authenticate an external party toward the A30. This key is written to the A30 with ManageCARootKey. This is further detailed in Section 6.7.2.

6.7.1 ECCPrivateKey Management

6.7.1.1 Command ManageKeyPair

The generation of ECCPrivateKeys is possible with the command ManageCARootKey.

A30 supports up to five ECC Key Pairs. The key pairs are associated with a specific curve via CurveID.

Each ECC key pair is assigned to a specific application area and potentially to a specific protocol via KeyPolicy. Typically, best security practice is to use each key for a single purpose. Therefore, if allowing multiple usages for the same key, the implication from security perspective must be assessed.

Generation of a new key pair requires the access condition and communication mode as defined in the configuration parameters (see SetConfiguration Option 0x12). By default, CommMode.Full is applied, requiring authentication granting AppMasterKey access rights. Key pair replacement requires write-access as specified with ECCPrivateKey was created.

During key pair generation, the private key is securely stored on-card and the public key is returned to the caller. In case of import, the private key is to provided via PrivateKey. In this case, the public key is not returned.

It is also possible to only update the metadata, i.e. KeyPolicy and access rights, of an existing key entry. This will not affect the current key value. For metadata update, also WriteAccess as configured for the targeted key entry is required. In this case, the command is rejected if the CurveID is not set to the curve associated with the current key.

6.7.1.2 ECCPrivateKey Key Usage Limit

To allow mitigating potential future attack scenarios, ECCPrivateKeys can be configured with a key usage limitation. This allows limiting the amount of private key computations, and therefore related trace collection for side-channel attacks. Next to attack mitigation, this feature can also be used to limit the usage of a card/device.

Potentially, the limit can be increased in the field, e.g. if the end user pays for additional service. The key usage limitation (KeyUsageCtrLimit) is configured through KUCLimit.

Once enabled for a particular ECCPrivateKey, any private key usage is counted through a KeyUsageCtr associated with that ECCPrivateKey. This means that the KeyUsageCtr shall be incremented by one before the private key operation of the following operations:

- ISOInternalAuthenticate for Card-Unilateral Authentication, see Section 6.3.3.
- ReadData or ISOReadBinary when applying Secure Dynamic Messaging with ECDSa SDMSIG, i.e.only when SDMSIG is targeted to be read out, see Section 6.3.8.10.
- CryptoRequest with action 0x03 for ECC signature generation, see Section 7.10.3. Note that in case of Initialize/Update/Finalize flow, the counter is incremented on the Finalize-step.
- CryptoRequest with action 0x05 for ECC Diffie-Hellman, see Section 7.10.5. Note that here the counter is only incremented in the Single-step flow, as the Tow-step flow does not support ECCPrivateKey.
- ISOGeneralAuthenticate for SIGMA-I, see Section 6.3.2:
  - A30 acting as Responder: before B1 response
  - A30 acting as Initiator: before A1 response

Note: *Any updates to the KeyUsageCtr are written with anti-tearing protection, guaranteeing that the counter will in case of tearing either hold the previous or the targeted value.*

If the configured KeyUsageCtrLimit has been reached, the related ECCPrivateKey will be disabled. This means that the key cannot be used for private key computations, though the key entry can still be updated (and potentially reenabled) if the required authentication to do so can still be gained. If the KeyUsageCtrLimit is disabled, private key operations are not counted.

When only updating metadata with ManageKeyPair, it is possible to disable or change the KeyUsageCtrLimit without affecting the current KeyUsageCtr value. Note that putting the limit to a value equal or lower than the current value, will immediately disable the key entry. When changing the current key value through an import or generate key pair action, the current KeyUsageCtr value shall be reset to zero.

It is also possible to freeze the current KeyUsageCtrLimit. This can be done through KeyPolicy Bit 15. Once the KeyUsageCtrLimit has been frozen, it cannot be updated anymore. This means that a ManageKeyPair updating metadata will be rejected if KUCLimit has a value different from the currently configured KeyUsageCtrLimit or if KeyPolicy Bit 15 is not set. Note that it is still possible to change the limit configuration by generating or importing a new key pair.

Enabling the key usage limit feature may create a denial-of-service risk. For typical use cases, the risk should be limited if e.g. configuring a limit of one million, or if preceding authentication of the external party is required before the A30 private key operation.

Note: *It is essential to properly protect the ECCPrivateKey write access, as the right to update the key entry also allows to update and/or disable the KeyUsageCtrLimit.*

6.7.1.3 ECCPrivateKey Information Retrieval

A30 supports information retrieval with regard to ECCPrivateKey by GetKeySettings as defined in Section 7.5.2.

A30 does not support exporting private keys or the related public keys. Note that the related public key is typically stored via a certificate in a FileType.StandardData file. If the certificate is not created at the time of ECCPrivateKey generation or import, the public key may be temporarily stored in the file and later overwritten with the certificate. Note that this means one needs to be careful when generating a key pair ManageKeyPair and putting the WriteAccess condition to 0xF. If the public key in the response gets lost, one is not able to regenerate the key entry. Therefore, it is not recommended to put WriteAccess to 0xF before the public key has been received.

6.7.2 CARootKey Management

6.7.2.1 Command ManageCARootKey

The writing of CARootKeys is possible with the command ManageCARootKey as defined in Section 7.6.2.

A30 supports up to five CARootKeys.

The public keys are associated with a specific curve via CurveID. Note that A30 does not validate the provided public key.

Each CARootKey has an associated set of access rights via AccessRights which can be granted to the host after successful authentication depending on the presented certificates. Note that AccessRights is encoded LSB first.

All CARootKeys can optionally be associated with a trusted issuer name via IssuerLen and Issuer. The full Issuer byte string, including SEQUENCE tag and length must be provided. If a trusted issuer name is set, this is compared against the Issuer field of the public key certificate provided during the authentication. In case of chaining, the (grand-)parent certificate Issuer must match. Note that the implementation stores a hash of the provided Issuer to allow for fixed memory consumption.

Creation of CARootKeys requires the access condition and communication mode as defined in the configuration parameters (see SetConfiguration Option 0x12). By default, CommMode.Full is applied, requiring authentication granting AppMasterKey access rights. Updating an existing CARootKey requires write-access as specified with WriteAccess when the entry was created.

If a certificate cache is enabled, see SetConfiguration Option 0x13, the cache will be flushed on updating a CARootKey.

6.7.2.2 CARootKey Information Retrieval

A30 supports information retrieval with regards to CARootKey by GetKeySettings as defined in Section 7.5.2.

6.7.3 PICC/MF level

6.7.3.1 ECCPrivateKey entries

At PICC or MF level, in the default configuration, the A30 is trust-provisioned during manufacturing with one key pair from which the private keys are stored on the A30 as ECCPrivateKeys for the purpose of originality checking. This is further detailed in Section 6.16.1.1.

6.7.4 Application/DF level

6.7.4.1 ECCPrivateKey entries

A30 supports up to five ECCPrivateKey entries.

6.7.4.2 CARootKey entries

A30 supports up to five CARootKey entries.

6.7.5 Memory Consumption

Memory allocation is done in 32-byte blocks, see Section 6.5.2.3.

The memory for asymmetric keys is allocated at their creation and is defined as follows:

- ECCPrivateKey: three blocks.
- CARootKey: four blocks.

6.7.6 Certificate Cache

The A30 supports a cache of validated public keys. This is used to accelerate protocol execution time by removing the need to validate public key certificates that have been previously verified. The cache uses a look-up mechansim, which allows a certificate to be validated if its parent has been previously verified. Use of the cache is controlled via a configuration option. When enabled, the cache is populated automatically by the A30 during protocol execution.

If no intermediate cache entry is located, then the A30 shall check for a matching root CA public key. If no entry is found, then verification shall be sequentially tried using all CA public key entries, which were loaded without associated issuer information.

The cache shall be partitioned into entries for end-leaf public-keys and entries for parent/grand-parent public keys i.e. public keys belonging to intermediate certificates. Each cache entry shall be stored with its expiry date. Although the A30 has no notion of the current time, it does keep track of the ‘latest time’. This is the most recent validity time from a validated certificate. The cache replacement scheme shall be ‘least recently used’; where the most recently used entries are retained. However, all expired certificates shall be flushed from the cache.

The size of the cache is determined by the CA Root Key cache configuration parameters. Table 30 illustrates a cache with 5 end-leaf slots and 2 intermediate certificate slots. The cache is created using the SetConfiguration command. The cache can only be created once and cannot be resized.

Table 30. Certificate Cache Example

End Leaf Certificates				Intermediate Certificates			
Cache Entry Num	Public Key Certificate Hash	Public Key	Expiry Date	Cache Entry Num	(Optional) CRC-16 of Certificate Subject Name	Public Key	Expiry Date
Slot 1				Slot 1			
Slot 2				Slot 2			
Slot 3							
Slot 4							
Slot 5							

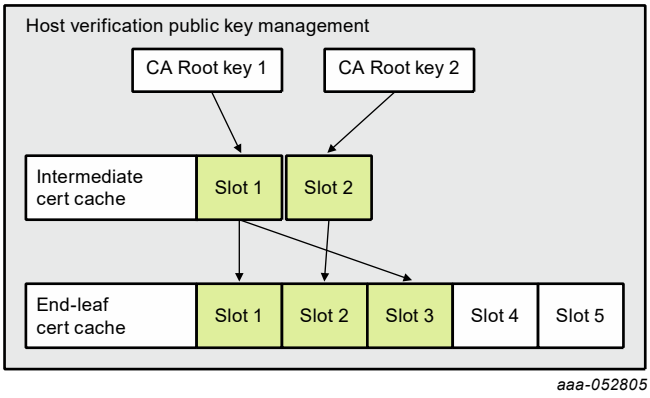


Figure 13. Conceptual View of Host Verification Public Keys

Figure 13 represents how the cache would be populated for a use case where A30 was authenticated with three different hosts with the hosts certificate chains as follows:

Host 1: leaf cert 1 -> intermediate cert 1 -> CA Root Key 1

Host 2: leaf cert 2 -> intermediate cert 2 -> CA Root Key 2

Host 3: leaf cert 3 -> intermediate cert 1 -> CA Root Key 1

6.8 Certificate Management

6.8.1 ECC Certificate Repository Management

The A30 supports certificate repositories. A certificate repository provides storage for the credentials required for the A30 to execute the SIGMA-I mutual authentication protocols. Construction of a certificate repository consists of the following steps:

- Create certificate repository. Note the maximum memory specified for the repository is allocated on creation. This size may be defined as larger than initially required to allow increasing data items after resetting a repository.
- Load one or more public key certificate
- Load one or more certificate mapping table (optional)
- Activate the certificate repository

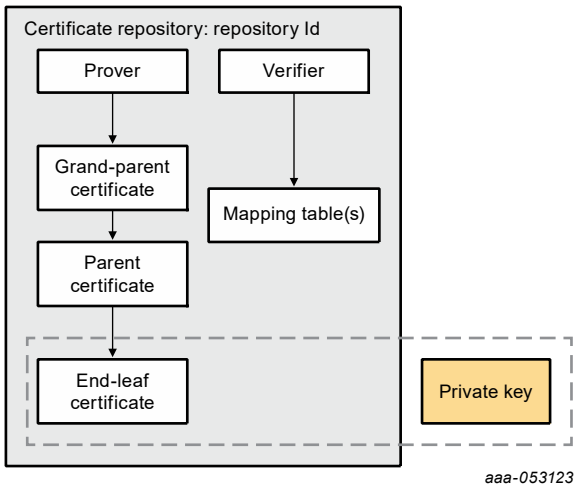


Figure 14. Conceptual View of a Certificate Repository

The certificate repository is populated and activated using the ManageCertRepo command outlined in Section 7.7.1. The access condition defined in the A30's configuration parameters is used for repository creation. The Read and Write/Reset access conditions provided during repository creation/reset otherwise apply. The command does not return any response data.

6.8.1.1 Create Certificate Repository

Creation of the certificate repository requires:

- the identity of the on-card private key to be associated with the repository
- a repository identifier used to personalize the repository and to access the repository during algorithm execution

The format of the create certificate repository command data is defined in Table 129.

6.8.1.2 Load Public Key Certificate Chain

The certificate chain shall include an end-leaf certificate and may optionally include up to two intermediate public key certificates. This enables support of a certificate chain four deep because the root CA public key (trusted root of the chain) is stored on the receiving entity (the verifier). Each certificate has its own public key and associated algorithm, therefore, chains may include a mix of algorithms.

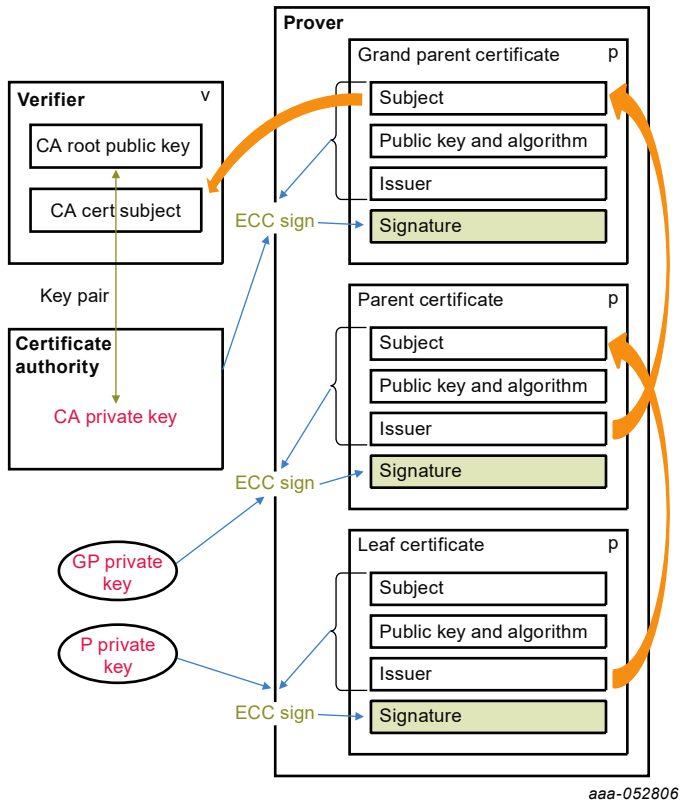


Figure 15. Certificate Chain Example

A separate command is required to load each certificate in the chain. The certificate repository supports loading either compressed or uncompressed certificates. If a compressed end-leaf certificate is loaded, then the hash of the associated uncompressed certificate also must be provided (this is required for SIGMA-I protocol execution). The command format is outlined in Table 130.

The A30 shall not verify the certificate chain or certificate hash values during loading.

6.8.1.3 Certificate Mapping Table

The A30 supports the X.509 certificate format for host certificates. This format has a defined certificate structure:

```
Certificate  ::=  SEQUENCE  {
            tbsCertificate       TBSCertificate,
            signatureAlgorithm   AlgorithmIdentifier,
            signatureValue       BIT STRING
            }

TBSCertificate  ::=  SEQUENCE  {
            version         [0]  EXPLICIT Version DEFAULT v1,
            serialNumber         CertificateSerialNumber,
            signature            AlgorithmIdentifier,
            issuer               Name,
            validity             Validity,
            subject              Name,
            subjectPublicKeyInfo SubjectPublicKeyInfo,
            issuerUniqueID  [1]  IMPLICIT UniqueIdentifier OPTIONAL,
            -- If present, version MUST be v2 or v3
            subjectUniqueID [2]  IMPLICIT UniqueIdentifier OPTIONAL,
            -- If present, version MUST be v2 or v3
```

```
            extensions      [3]  EXPLICIT Extensions OPTIONAL
            -- If present, version MUST be v3
            }

AlgorithmIdentifier ::= SEQUENCE { algorithm OBJECT IDENTIFIER, parameters ANY DEFINED BY
 algorithm OPTIONAL }
```

6.8.1.3.1 x.509 Wrapping

The A30 allows a certificate wrapping to be defined, e.g., PKCS#7. The wrapping basically provides a path, using ASN.1 encoding, to the start of the x.509 certificate, Providing an x.509 wrapping path is optional. If it is not provided, then the A30 assumes the x.509 certificate is not wrapped. Wrapping information is loaded using tag ‘A0’ see Table 31. Following is a wrapping example using CMS/PKSC#7 format:

```
    <SEQUENCE> (0x30) [0xyyyy]
    {
        <OID> (0x06) [0x09] { 2A864886F70D010702 } -> iso(1) member-body(2)
         us(840) rsadsi(113549) pkcs(1) pkcs7(7) 2

        <CONTEXT SENSITIVE> (0xA0) [0xyyyy]
        {
            <SEQUENCE> (0x30) [0xyyyy]
            {
                <INTEGER> (0x02) [0x01] { 01 } -> version
                <SET> (0x31) [0x00] -> set of DigestAlgorithmIdentifier
                <SEQUENCE> (0x30) [0x0B]
                {
                    <OID> (0x06) [0x09] { 2A864886F70D010701 } ->iso(1) member-body(2)
         us(840) rsadsi(113549) pkcs(1) pkcs7(7) 1
                }
                <CONTEXT SENSITIVE> (0xA0) [0xyyyy]
                {
          //x.509 certificate which starts with <SEQUENCE> (0x30)
```

Table 31. X.509 Certificate Wrap Encoding

Tag	Description
‘A0’	The ASN path to the start of the full x.509 certificate e.g. A0 0E 3081A08130810282318230 82A083 for the PKSC#7 wrapping outlined above The ASN path consists of pairs of tag|qualifier path entries where the qualifier is one of the following: 0x81 The next ASN tag is nested inside the current element 0x82 The next ASN tag is at the same level as the current element 0x83 End of list ***Note:*** *The first entry matching the tag provided is located, therefore, multiple path entries may be required when there are duplicate tags*

6.8.1.3.2 Mapping Table Command Data Format

The format of the command data required to load a certificate mapping table is outlined in Table 131

6.8.1.4 Activate Certificate Repository

Once personalized, the certificate repository must be activated. The format of the certificate repository activation command data is defined in Table 128.

6.8.2 Read Certificate Repository

It is possible to read a certificate from a repository or to read a repository's metadata using the ReadCertRepo command. Reading metadata does not require any authentication; if reading metadata in a secure tunnel then CommMode.MAC is applied. Reading a certificate directly from the repository requires access as defined in the Read access condition set during repository creation/reset. If reading using a standard APDU then the maximum response data length is 239 bytes. The format of the ReadCertRepo command is defined in Section 7.7.2.

6.9 Application Management

A30 groups user data into an application. Within an application data is further grouped into files, as described in Section 6.10.

A30 only holds one application, which is pre-configured at delivery, as defined in Section 6.9.2.

In Section 6.9.1, it is detailed how applications can be selected.

6.9.1 Application Selection

An application can only be selected with ISOSelectFile, see Section 6.15.1.4.

6.9.2 Application Definition

A30 comes pre-configured with one application. It shall have the following properties for application selection:

- DFName: 0xD2760000850101
- ISOFile Identifier: 0xE110

6.10 File Management

A30 maintains user data into files of specific types listed in Section 6.10.1. Files are managed through creation, information retrieval and update functions respectively specified in Section 6.10.4, Section 6.10.3 and Section 6.10.2.3. File access can be restricted with an access right management specified in Section 6.10.2.

6.10.1 File Types

A30 supports the following types of data storage:

- raw data as specified in Section 6.10.1.1
- monotonic counters as specified in Section 6.10.1.2

All A30 files are defined with a file number and the communication mode that has to be used when accessing the file data. The file number is coded over 1 byte. It is unique per file in an application. The communication mode is defined in Section 6.3.6.6.

6.10.1.1 FileType.StandardData

FileType.StandardData stores the data as raw data byte per byte. Data is accessed by chunk of byte at a certain offset in the data file and with a certain length in byte.

A FileType.StandardData file is created with CreateStdDataFile, see Section 6.10.4.1. As defined in Section 6.10.6, A30 holds three FileType.StandardData files at delivery. Next to this, the user can create additional files.

A FileType.StandardData file can be read with ReadData and ISOReadBinary. The data can be written with WriteData and ISOUpdateBinary.

FileType.StandardData is defined by its size in bytes. The size of each of the additional files the user can create, is limited to maximum of 1024 bytes.

Limited anti-tearing protection is foreseen, as it is ensured that the data received in a single frame is written anti-tearing protected, i.e. all targeted data or none of it is updated. In case of chaining, see Section 6.2.3, an A30 buffers multiple frames up to the supported FSC size of 256 bytes and write them at once. Note that in this case, if secure messaging applies, incomplete cryptographic blocks within a frame cannot be fully processed. Such a block will then be considered as part of the next frame.

6.10.1.2 FileType.Counter

FileType.Counter stores a 4-byte monotonic counter. This means that the counter can only be incremented and never decremented.

A FileType.Counter file is created with IncrementCounterFile, see Section 6.10.4.2. As defined in Section 6.10.6, A30 does not hold any FileType.Counter files in the default configuration at delivery.

One of the counters can be enabled as Authentication counter with SetConfiguration Option 0x16. The Authentication counter is incremented every time a symmetric and/or asymmetric mutual authentication session is initiated. It can also be incremented with IncrementCounterFile. Note that only one FileType.Counter can have this function. For more details, also on configuring a limit on the number of allowed authententications, see Section 6.3.5.

The other User Counters can be incremented only with IncrementCounterFile. It is not possible to update counters using WriteData or ISOUpdateBinary.

Any FileType.Counter file can be read with GetFileCounters. IncrementCounterFile has built in anti-tearing protection, guaranteeing that the counter will in case of tearing either hold the previous or the incremented value.

6.10.2 File Access Rights Management

For a generic introduction on access right management, see Section 6.4 and especially Section 6.4.1 for the encoding of access conditions.

File data is accessed with three different access rights: FileAR.Read, FileAR.Write and FileAR.ReadWrite. Each of these access rights are permitting the use of a subset of commands defined in Section 6.10.2.2.

In addition, an access right called FileAR.Change is specified per file permitting ChangeFileSettings to change the file access rights.

An access right is granted if at least one condition associated to it is satisfied. Such conditions are called access conditions.

The set of access conditions are coded on 2 bytes as shown in Table 32. RFU access conditions are expected to be set to 0xF (for future extensibility).

Table 32. Set of Access condition coded on 2 bytes

Bit index	Description	Value
15..12	FileAR.Read	access condition as in Table 17.
11..8	FileAR.Write	access condition as in Table 17.
7..4	FileAR.ReadWrite	access condition as in Table 17.
3..0	FileAR.Change	access condition as in Table 17.

6.10.2.1 Secure Dynamic Messaging Related Access Rights

Additionally, a FileType.StandardData file can be associated with the following Secure Dynamic Messaging access rights:FileAR.SDMMetaRead, FileAR.SDMFileRead, and FileAR.SDMCtrRet.

The FileAR.SDMCtrRet is interpreted as the access rights defined above, according to Table 17 and grants access to GetFileCounters. The others have a different interpretation.

The FileAR.SDMMetaRead is a bit special as it does not define access to certain commands, i.e.by setting this access right one does not affect the policy on when certain commands will be allowed or not. It purely defines the mirroring of PICCData, i.e. whether the PICCData will be mirrored in plain, encrypted or not at all, see also Section 6.3.8.3. This is interpreted according to Table 33.

Table 33. FileAR.SDMMetaRead values

Condition value	Description
0x0..0x4	SDMMetaReadKey: key number of an AppKey used to encrypt the PICCData before mirroring
0xE	Plain PICCData mirroring
0xF	No PICCData mirroring

The FileAR.SDMFileRead and FileAR.SDMFileRead2 will, as soon as one of them is different from 0xF, grant free access to ReadData and ISOReadBinary.

The FileAR.SDMFileRead, as defined in Table 34, allows configuring a symmetric AppKey. This key is used to derive session keys, see Section 6.3.8.12. SesSDMFileReadMACKey is used for SDMMAC computation as defined in Section 6.3.8.8 and Section 6.3.8.9. SesSDMFileReadENCKey is used for file data encryption. See SDMENCFileData as defined in Section 6.3.8.6 and Section 6.3.8.7.

The FileAR.SDMFileRead2, as defined in Table 34, allows configuring an asymmetric ECCPrivateKey. This key is used for SDMSIG computation as defined in Section 6.3.8.10 and Section 6.3.8.11. If both FileAR.SDMFileRead andFileAR.SDMFileRead2 configure a key, an SDMSIG is computed with the key of FileAR.SDMFileRead2. No SDMMAC is calculated in this case, but FileAR.SDMFileRead will still be used for encryption if enabled. Table 35 gives an overview of the possible combinations.

Table 34. FileAR.SDMFileRead values

Condition value	Description
0x0..0x4	SDMFileReadKey: free access, key number of an AppKey that is to be applied for the Secure Dynamic Messaging
0xE	RFU
0xF	No symmetric Secure Dynamic Messaging for Reading

Table 35. FileAR.SDMFileRead2 values

Condition value	Description
0x0..0x4	SDMFileReadKey: free access, key number of an ECCPrivateKey that is to be applied for the SDMSIG calculation
0xE	RFU
0xF	No asymmetric Secure Dynamic Messaging for Reading

Table 36. FileAR.SDMFileRead and FileAR.SDMFileRead2 combinations

FileAR.SDMFileRead	FileAR.SDMFileRead2	SDMENCFileData	SDMMAC	SDMSIG	Comment
AppKey	valid ECCPrivateKey	Yes, mandatory to be enabled	No	Yes	-
AppKey	invalid ECCPrivateKey	Yes, mandatory to be enabled	No	No	Rejected at ChangeFileSettings. If ECCPrivateKey gets invalidated afterward, the static file data is returned at SDMMACOffset.
AppKey	0xF	Yes, if enabled	Yes	No	-
0xF	valid ECCPrivateKey	No	No	Yes	-
0xF	invalid ECCPrivateKey	No	No	No	Rejected at ChangeFileSettings. If ECCPrivateKey gets invalidated afterward, the static file data is returned at SDMMACOffset.
0xF	0xF	No	No	No	-

6.10.2.2 Access right association with commands

In Table 37, it is listed to which commands the access rights are granting access to.

Table 37. Command list associated with access rights

AccessRight	Commands
FileAR.Read	ReadData ISOReadBinary GetFileCounters if targeting FileType.Counter
FileAR.Write	WriteData ISOUpdateBinary IncrementCounterFile
FileAR.ReadWrite	ReadData WriteData ISOReadBinary ISOUpdateBinary GetFileCounters if targeting FileType.Counter IncrementCounterFile
FileAR.Change	ChangeFileSettings
FileAR.SDMMetaRead	-

Table 37. Command list associated with access rights*...continued*

FileAR.SDMFileRead or **FileAR.SDMFileRead2**	ReadData ISOReadBinary
FileAR.SDMCtrRet	GetFileCounters if targeting FileType.StandardData

A command listed in Table 37 is accepted if at least one access condition associated with an access right (could be several) granting access to it is satisfied. If authenticated and the only access conditions satisfied are free access 0xE within FileAR.Read, FileAR.Write, FileAR.ReadWrite and FileAR.Change, then the CommMode.Plain is to be applied.

If not authenticated, Secure Dynamic Messaging will be applied if access is granted via FileAR.SDMFileRead or FileAR.SDMFileRead2, even if there is free access via one of the other access rights. FileAR.SDMFileRead and FileAR.SDMFileRead2 are not affecting the regular secure messaging, i.e. if authenticated.

Note: *GetFileCounters access is only granted via FileAR.Read andFileAR.ReadWrite if targeting a FileType.Counter. If targeting a FileType.StandardData, access to GetFileCounters is only granted via the dedicated Secure Dynamic Messaging FileAR.SDMCtrRet.*

A command listed in Table 37 is rejected if there is no satisfied access conditions associated with an access right (could be several) granting access to it. The command returns:

- Resp.PERMISSION_DENIED if all access conditions associated with all access rights granting access to the command are denying any access.
- Resp.AUTHENTICATION_ERROR if at least one access condition associated with one of the access rights granting access to the command requires a valid authentication, while being in VCState.NotAuthenticated, or in VCState.AuthenticatedAES but authenticated with the wrong key.
- Resp.CERT_ERROR if at least one access condition associated with one of the access rights granting access to the command requires a valid authentication, while being in VCState.AuthenticatedECC but not having obtained the required access rights from the targeted CARootKey or reader certificate presented during the authentication.

6.10.2.3 Command ChangeFileSettings

ChangeFileSettings as defined in Section 7.8.7 permits to update the communication mode of a file as specified in Table 14 and the access rights of a file by mean of all its sets of access conditions as specified in Table 32.

The *AccessRights* parameter is mandatory and updates the mandatory set of access conditions and defined in Table 32.

ChangeFileSettings also allows enabling the Secure Dynamic Messaging and mirroring features, see Section 6.3.8 for more details. Note that it is possible to defer the encryption configurations of the SDM configuration, see Deferred Configuration Options.

If targeting a FileType.Counter, the counter can be enabled as Authentication Counter by setting *FileOption* Bit 6. If another file is currently already enabled as Authentication Counter, the feature will be disabled for the previous file, i.e. only one FileType.Counter can act as Authentication Counter at a time.

The command is rejected if:

- one of the access conditions is targeting a key that does not exist within the application.
- the PICC level is selected.
- the *FileNo* parameter does not refer to an existing file in the selected application.
- the FileAR.Change is not granted because it is a no access 0xF.
- the FileAR.Change is not granted because it requires an authentication with a AppKey which is currently not active.
- trying to enable Secure Dynamic Messaging on a file where it is not supported.
- the provided configuration for Secure Dynamic Messaging and mirroring is inconsistent according to the conditions of Section 6.3.8, as reflected in Section 7.8.7.

Under active authentication, the command ChangeFileSettings requires CommMode.Full. There is one exception: if FileAR.Change of the targeted file is configured to 0xE allowing free access, also under active authentication CommMode.Plain is to be applied. Information on authentication and secure messaging-dependent structure of the command can be found in Section 6.3.

6.10.3 File Information Retrieval

6.10.3.1 Command GetFileSettings

GetFileSettings as defined in Section 7.8.5 allows to get information on the properties of a specific file. The information provided by this command depends on the type of the file which is queried.

The file from which the settings have to be retrieved is defined by FileNo specified over 5 bits. The first part of the returned message is the same for all file types:

- the actual file type, see Section 6.10.1
- the communication mode as specified in Table 14
- the access rights of a file by mean of all its sets of access conditions as specified in Table 32.

All subsequent bytes in the response have a special meaning depending on the file type:

- FileType.StandardData: file size over 3 bytes. If Secure Dynamic Messaging, with eventually Deferred Configuration, applies for the targeted file, this is also indicated, and the related parameters are returned.
- FileType.Counter file: if the authentication Counter is enabled. The command is rejected if:
- the targeted file does not exist

The command is rejected if:

- the targeted file does not exist

Under active authentication, the command GetFileSettings requires CommMode.MAC. Information on authentication and secure messaging-dependent structure of the command can be found in Section 6.3.

6.10.3.2 Command GetFileCounters

GetFileCounters as defined in Section 7.8.6 supports retrieving of the following counter values:

- current values associated with the 24-bit SDMReadCtr related with a FileType.StandardData file after enabling Secure Dynamic Messaging, see Section 6.3.8 and Section 6.10.2.3.
- current values associated with the FileType.Counter files holding a 32-bit counter.

The command is rejected if

- The PICC level is selected
- the targeted file does not exist
- the targeted file is not a FileType.StandardData file with Secure Dynamic Messaging enabled, or a FileType.Counter file.
- if targeting FileType.StandardData file, depending on FileAR.SDMCtrRet, permission is always denied or requires authentication.
- if targeting FileType.Counter file, depending on FileAR.Read or FileAR.ReadWrite, permission is always denied or requires authentication.

Under active authentication, the command GetFileCounters requires CommMode.Full for SDMReadCtr retrieval. If retrieving the value of a FileType.Counter, the communication mode depends on the configuration of the file. Information on authentication and secure messaging-dependent structure of the command can be found in Section 6.3.

6.10.3.3 Command GetFileIDs

GetFileIDs as defined in Section 7.8.3 returns the complete list of file IDs of all active files of the selected application.

The command takes no parameters.

Each File ID is coded in one byte. Duplicate values are not possible as each file must have an unambiguous identifier.

The response includes all identifiers of all FileType.StandardData or FileType.Counter files. For FileType.StandardData, independently of whether they were pre-allocated or created by the user.

The command is rejected if:

- the PICC level is selected.

Under active authentication, the command GetFileIDs requires CommMode.MAC. Information on authentication and secure messaging-dependent structure of the command can be found in Section 6.3.

6.10.3.4 Command GetISOFileIDs

GetISOFileIDs as defined in Section 7.8.4 returns the complete list of the 2 byte ISO/IEC 7816-4 File IDentifiers of all active files within the currently selected application.

The command takes no parameters.

Each File ID is coded in 2 bytes. Duplicate values are not possible as each file must have an unambiguous identifier.

The response includes all identifiers of all FileType.StandardData files, independently of whether they were pre-allocated or created by the user.

The command is rejected if:

- the PICC level is selected.

Under active authentication, the command GetISOFileIDs requires CommMode.MAC. Information on authentication and secure messaging-dependent structure of the command can be found in Section 6.3.

6.10.4 File Creation

A30 supports file creation for FileType.StandardData and FileType.Counter files.

The file creation commands all share the following parameters: *FileNo*, *FileOption* and *AccessRights*.

The *FileNo* encodes the file number in the range of 0x00 to 0x1F which the new created file should get within the currently selected application. If the file number is already occupied, the file creation fails.

FileOption defines the communication mode of the targeted file, see Section 6.3.6.6.

The *AccessRights* define the mandatory access right set of the newly created file. Note that the meaning of these access rights depends on the targeted file type, see Section 6.10.2.2. The command is rejected if one of the access rights targets a key that is not available in the targeted application.

The file creation command is rejected if no application has been selected, i.e. the PICC level is currently selected. An active authentication with the AppMasterKey is required.

Under active authentication file creation commands require CommMode.MAC.

6.10.4.1 Command CreateStdDataFile

General aspects of file creation, shared by all file creation commands, are described at the start of Section 6.10.4. In addition to the parameters listed above, CreateStdDataFile, as defined in Section 7.8.1, specifies the size of the file in bytes. FileSize is defined as a 3 byte integer. The file will be initialized with all zero bytes.

Every FileType.StandardData file within the application, must be created with a 2 byte File Identifier *ISOFileID* to enable ISO/IEC 7816-4 selection with ISOSelectFile.

A30 does not limit the amount of files that can be created, other than by the available memory and FileNo range.

The size of a created file must not exceed 1024 byte.

6.10.4.2 Command CreateCounterFile

General aspects of file creation, shared by all file creation commands, are described at the start of Section 6.10.4. In addition to the parameters listed above, CreateCounterFile, as defined in Section 7.8.2, specifies the initial value of the counter. Value is defined as a 4 byte unsigned integer.

If targeting a FileType.Counter, the counter can be enabled as Authentication Counter by setting FileOption Bit 6. If another file is currently already enabled as Authentication Counter, the feature will be disabled for the previous file, i.e. only one FileType.Counter can act as Authentication Counter at a time.

A30 does not limit the amount of counters that can be created, other than by the available memory and FileNo range.

6.10.5 Memory Consumption

Memory allocation is done in 32-byte blocks, see Section 6.5.2.3.

The memory for files is allocated at file creation and can be computed as follows:

- General overhead: 1 block per 2 files within an application.
- FileType.StandardData: (*FileSize* +31)/32
- FileType.Counter: 1 block.

6.10.6 File Definition

The A30 application as defined in Section 6.9.2 shall hold the following files:

FileType.StandardData files

- a FileType.StandardData file of 32 bytes with following properties:
  - FileNo = 0x01; ISO File ID = 0xE103
  - FileAR.Read = 0xE;FileAR.Write =0x0;FileAR.ReadWrite =0x0;FileAR.Change =0x0
  - Secure Dynamic Messaging and mirroring are not supported for this file.
  - CommMode.Plain

This file will hold the CC-file according to [14]. At delivery it will hold following content:

- CCLEN = 0x0017, i.e. 23 bytes
- T4T_VNo = 0x20, i.e. Mapping Version 2.0
- MLe = 0x0100, i.e. 256 bytes
- MLc = 0x00FF, i.e. 255 bytes
- NDEF-File_Ctrl_TLV
  - T = 0x04, indicates the NDEF-File_Ctrl_TLV
  - L = 0x06, i.e. 6 bytes
  - NDEF-File File Identifier = 0xE104
  - NDEF-File File Size = 0x0100, i.e. 256 bytes
  - NDEF-File READ Access Condition = 0x00, i.e. READ access granted without any security
  - NDEF-File WRITE Access Condition = 0x00, i.e. WRITE access granted without any security
- Proprietary-File_Ctrl_TLV
  - T= 0x05, indicates the Proprietary-File_Ctrl_TLV
  - L= 0x06, i.e. 6 bytes
  - Proprietary-File File Identifier = 0xE105
  - Proprietary-File File Size = 0x0080, i.e. 128 bytes
  - Proprietary-File READ Access Condition = 0x82, i.e. Limited READ access, granted based on proprietary methods, after authentication with key 0x2.
  - Proprietary-File WRITE Access Condition = 0x83, i.e. Limited READWRITE access, granted based on proprietary methods, after authentication with key 0x3.

The remainder of the file is set to all 0x00 bytes.

- a FileType.StandardData file of 256 bytes with following properties:
  - FileNo= 0x02; ISO File ID = 0xE104
  - FileAR.Read =0xE;FileAR.Write =0xE;FileAR.ReadWrite =0xE;FileAR.Change =0x0
  - SecureDynamic Messaging and mirroring is supported for this file, but disabled at delivery.
  - CommMode.Plain
  - By default, this file is set to all 0x00 bytes at delivery. This file will hold the NDEF-file according to [14].
- a FileType.StandardData file of 128 bytes with following properties:
  - FileNo= 0x03; ISO File ID = 0xE105
  - FileAR.Read =0x2;FileAR.Write =0x3;FileAR.ReadWrite =0x3;FileAR.Change =0x0
  - SecureDynamic Messaging and mirroring is not supported for this file.
  - CommMode.Full
  - By default, this file is set to 0x00 0x7E, followed by all 0x00 bytes at delivery.

This file proprietary file according to [14] that can hold additional confidential information. According to [14], the PLEN field is set to 126 (0x007E) by default at delivery.

All files can on request get customer-specific configurations and contents through commercial customization options, instead of the default values listed here.

After personalization the write access to the FileType.StandardData files, can be adapted to no access (0xF).

The following access rights for Secure Dynamic Messaging can be configured by the customer, e.g. as follows:FileAR.SDMMetaRead = 0x4;FileAR.SDMFileRead = 0x1;FileAR.SDMCtrRet = 0x2. This is only a recommended setting, other configurations are also possible. In this setting KeyNo 0x4 is used as non-diversified key (e.g. in this case configuring for encrypted UID-retrieval via PICCData). KeyNo 0x1 is used as read key protecting the file communication and KeyNo 0x2 is used for counter retrieval after mutual authentication.

6.11 Data Management

A30 maintains user data into files of specific types as described in Section 6.10. The user can access and manage the data through functions specific to file type.

Data can be read, written, or updated. Depending on the file type, data are defined as:

- raw data in FileType.StandardData

For a user, the access to data is limited by the access rights set at file level as defined in Section 6.10.2 and listed in Table 37.

6.11.1 Standard Data Files

6.11.1.1 Command ReadData

Reading data from FileType.StandardData files is possible with the command as defined in Section 7.9.1.

The data to be read is defined by the file number of the targeted file, the offset in the data file where to start the reading and its size in bytes. The file number specifying the file where to read the data from is given by FileNo specified over 5 bits as defined in Section 6.10.

The position byte-wise in the data file where to start to read data is given by Offset. Its valid range is from 0*x*000000 to *FileSize* −1. The data size to be read is given by Length specifying the number of bytes. If Length is equal to 0*x*000000 then the entire data file has to be read starting from the position specified by the Offset value. Length valid range is 0*x*000000 to *FileSize − Offset*.

Note: *Due to the ISO/IEC 7816-4 wrapping, only supporting short Le, see Section 6.2.2, the amount of data read is limited by Le as well.*

The data is returned in Data. If the number of bytes to send to the PCD does not fit into one single frame, chaining is applied, see Section 6.2.3.

As listed in Table 37, ReadData is allowed only if at least one of FileAR.Read and FileAR.ReadWrite access rights associated with the targeted file is granted.

Additionally, if not authenticated, ReadData may be granted if Secure Dynamic Messaging for reading is enabled via FileAR.SDMFileRead.

If authenticated, the communication mode depends on the one from the file being accessed as specified in Section 6.3.6.6. Information on authentication and secure messaging-dependent structure of the command can be found in Section 6.3.

6.11.1.2 Command WriteData

Writing data to FileType.StandardData files is possible with the command as defined in Section 7.9.2.

The location of data to be written is defined by the file number of the targeted file, the offset in the data file where to start the writing and its size in bytes. The file number specifying the file where to write to is given by FileNo specified over 5 bits as defined in Section 6.10.

The position byte-wise in the data file where to start to write data is given by Offset define on 3 bytes. Its valid range is from 0*x*000000 to *FileSize* −1. The data size to be written is given by Length specifying the number of bytes defined on 3 bytes. Length valid range is 0*x*000001 to *FileSize* −*Offset*. The data is passed in Data and is, if needed, split in multiple frames depending on the command variant as defined above.

The FileType.StandardData does offer limited anti-tearing protection, see Section 6.10.1.1.

For FileType.StandardData, data written in a file can be directly returned with ReadData, as FileType.StandardData does not implements any backup mechanism. Note especially that in case of chaining, data is already written before the integrity has been checked (CommMode.MAC or CommMode.Full). Therefore, in case of Resp.INTEGRITY_ERROR, the content of the file can be corrupted. For this reason, chained writing to FileType.StandardData in CommMode.MAC or CommMode.Full can be disabled with SetConfiguration, option 0x04. Note however, that also here an implementation may buffer multiple chained frames and write them at once. As long as the implementation can guarantee that the MAC is validated before the writing and all targeted data or none are updated, this does not violate the disabled chained writing configuration.

As listed in Table 37, WriteData is allowed only if at least one of FileAR.Write and FileAR.ReadWrite access rights associated with the targeted file is granted.

The communication mode depends on the one from the file being accessed as specified in Section 6.3.6.6. Information on authentication and secure messaging-dependent structure of the command can be found in Section 6.3.

At PICC level, the command is rejected.

6.11.2 Counter Files

6.11.2.1 Command IncrementCounterFile

Increment the value of a FileType.Counter file is possible with the command IncrementCounterFile as defined in Section 7.9.3.

The increment is defined by the file number of the targeted FileType.Counter file and the amount to add up. The file number specifying the file where to credit the amount is given by FileNo specified over 5 bits as defined in Section 6.10.

The increment amount is given in IncrValue over 4 bytes defined as an unsigned integer.

As listed in Table 37, IncrementCounterFile is allowed only if at least one of FileAR.Write or FileAR.ReadWrite access rights associated with the targeted file is granted.

The communication mode depends on the one from the file being accessed as specified in Section 6.3.6.6. Information on authentication and secure messaging-dependent structure of the command can be found in Section 6.3.

At PICC level, the command is rejected.

6.12 Crypto API

The A30 supports execution of crypto primitives via the CryptoRequest command. The crypto API enables execution of the following crypto operations:

- Random Number Generation [21][22]
- SHA-256/SHA-384[17]
- ECC Sign/Verify [23]
- ECC Diffie-Hellman [27]
- AES CMAC (128-bit and 256-bit key size) [7]
- AES CBC (128-bit and 256-bit key size) [5][6]
- AES ECB (128-bit and 256-bit key size) [5][6]
- AES CCM (128-bit and 256-bit key size) [25]
- AES GCM (128-bit and 256-bit key size) [26]
- Write to Internal Buffer storage
- HMAC [24]
- HKDF [28]
- Echo

The crypto API provides two internal buffers, which can be used as workspace for RNG data, keys, ECDH output, signature generation/verification and AES encrypt/decrypt. The buffers may be used as 1 or more 16-byte buffers as outlined in Figure 16 and Figure 17. One buffer provides data only retained for the current crypto API session (the transient buffer); the other buffer stores data persistently in NVM (the static buffer).

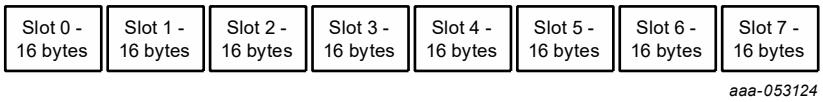


Figure 16. Crypto API Transient Buffer Format

The transient buffer is initialized (all zeroes) on the first Crypto API request following a Cold reset. The transient buffer is reinitialized under the following circumstances.

1. Warm reset,
2. ISO GeneralAuthenticate command
3. AuthenticateEV2First/NonFirst commands
4. Following an update to the Crypto API configuration (option 0x15).

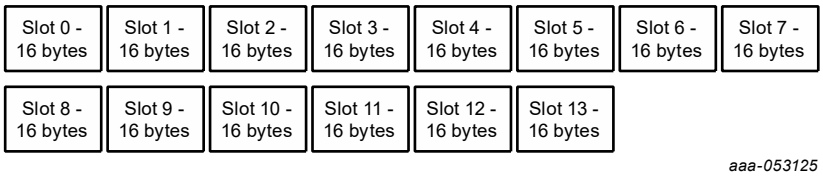


Figure 17. Crypto API Static Buffer Format

The initial state of the static buffer are all zeroes. The contents of the static buffer can be set using the Crypto API functions. The OS is never implicitly clear or reset the static buffer contents. The contents of the static buffer are stored securely by ciphering, and integrity protecting the contents when data is written to the buffer. This is done implicitly by A30.

The API permits selection of the input data source and cryptographic keys (if applicable). Keys may be ‘crypto API’ keys stored statically in the A30 or keys stored in a crypto API internal buffer. It is also possible to select the destination for the algorithm result. Input/output destination is selected in accordance with Table 38. If the

number of input or output data bytes exceeds the slot size, then the next slot is used for example, targeting an SHA operation to slot 0 will cause data to be written to both slots 0 and 1.

Table 38. Crypto API Data Source/Destination Selection

b7	b6	b5	b4	b3	b2	b1	b0	Description
0	0	0	0	0	0	0	0	Command buffer
1	0	0	0	0	Slot Num			Transient buffer slot number (0 to 7)
1	1	0	0	Slot Num				Static buffer slot number (0 to 15)

The usage of an internal buffer slot can be restricted using a policy option. The policy values are taken from the OS configuration area and are set using the SetConfiguration command. If no policy is set, then full access is permitted. If a command uses multiple slots, then the policy checks for each slot must be fulfilled.

Table 39. Crypto API Slot Usage Policy Options

b7	b6	b5	b4	b3	b2	b1	b0	Description
-	-	-	-	-	-	-	x	Can be used as input data for algorithms specified 0: disabled 1: enabled
-	-	-	-	-	-	x	-	Can be used as a key with algorithms specified 0: disabled 1: enabled

Table 40. Crypto API Policy Supported Algorithms

b7	b6	b5	b4	b3	b2	b1	b0	Description
-	-	-	-	-	-	-	x	HMAC 0: disabled 1: enabled
-	-	-	-	-	-	x	-	HKDF 0: disabled 1: enabled
-	-	-	-	-	x	-	-	SHA 0: disabled 1: enabled
-	-	-	-	x	-	-	-	AES 0: disabled 1: enabled
-	-	-	x	-	-	-	-	ECC DSA 0: disabled 1: enabled

The CryptoRequest command format is outlined in Table 180. It requires the command access defined in the configuration. Only a single crypto operation is supported for example, if a multipart SHA operation is initiated and then a request is received to execute an AES operation then the SHA operation shall be aborted.

Note: *Due to the maximum Lc value being 255, this restricts the maximum amount of input data for each APDU.*

If an internal buffer is referenced as input or output, then multiple slots are used for values of more than 16 bytes.

6.13 GPIO Management

A30 supports two GPIOs:

- GPIO1 may be configured for input detection of a binary-state input signal (e.g. to detect if button is pressed or not), binary-state output signal or down-stream power-out.
- GPIO2 may be configured as a binary-state input or output signal.

GPIO configuration is done with SetConfiguration 0x11, see Command SetConfiguration, and especially Table 1. For each of the modes, dedicated HW aspects as outlined in Table 3 can be set. When configured for down-stream power-out, a targeted voltage/current level needs to be set. It is however possible to overwrite this at runtime via ManageGPIO, if e.g. not sufficient power can be harvested from the actual field strength. When configured as output, the GPIO can also be configured to notify on authentication. This is further detailed in Section 6.13.4.

The ReadGPIO command, as defined in Section 7.11.2, may be used to read the current status of the GPIOs. The ManageGPIO, as defined in ManageGPIO, is used for controlling the output on GPIO1.

6.13.1 Command ManageGPIO

The ManageGPIO command format is outlined in Section 6.13.

The command is only accepted at application level, and will be rejected at PICC level. Depending on the ManageGPIOAccessCondition, as configured with SetConfiguration Option 0x11 , the command may require an authentication and the configured secure messaging communication mode. By default, the command is disabled.

When a GPIO is configured for output, after a power-on reset, the GPIO will be initialized with the state as configured by the GPIOXConfig from SetConfiguration Option 0x11 on the first command after activation, i.e. I^2C activation. This is independently of the state from a previous activation. Immediately after the PoR, output GPIOs will be in high-impedance (High-Z) state.

6.13.2 Command ReadGPIO

The ReadGPIO, as defined in Section 7.8.2, returns the status of GPIO1 and/or GPIO2 for both input and output use cases, as configured with SetConfiguration Option 0x11, see Command SetConfiguration.

For GPIO input configurations (GPIOXMode = 0x01), only the current status for GPIO1 and GPIO2 are returned:GPIO1CurrStatus and GPIO1CurrStatus. This is the value as measured during the execution of the ReadGPIO command.

For GPIO output (GPIOXMode = 0x02 or GPIO2Mode = 0x05) and down-stream power out (GPIO1Mode = 0x04) operations, the current status can be retrieved, i.e. whether or not the output or down-stream power out has been set or not. This allows an external host to keep track.

For both input and output cases, GPIO1CurrStatus and GPIO1CurrStatus can take the following values:

- *Low*: 0x4C, i.e. ASCII encoding of ‘L’. This is the value for a logical ‘0’, e.g. if the button is not pressed. In case of output, the output is not driven, or down-stream power out is not enabled (e.g. after CLEAR operation with ManageGPIO).
- *High*: 0x48, i.e. ASCII encoding of ‘H’. This is the value for a logical ‘1’, e.g. if the button is pressed. In case of output, the output is driven or down-stream power out is enabled (e.g. after SET operation with ManageGPIO).

Note: *All output cases for a GPIO pin configuration are covered in a single row in the table below. The following value is returned if the GPIO pin is disabled, or the feature is not enabled yet:*

- *Invalid*: 0x49,i.e. ASCII encoding of 'I'. This is the value when the feature has not been enabled. The complete GPIO status is returned on 3 bytes:
- Byte[0]:TTPermStatus or N/A
- Byte[1]:TTCurrStatus, GPIO1CurrStatus or N/A
- Byte[2]:GPIO1CurrStatus or N/A

This results in the following possible outputs, depending on the GPIO configuration:

Table 41. ReadGPIO response

Configuration		Response data		
GPIO1Conf	**GPIO2Conf**	**GPIOByte0**	**GPIOByte1**	**GPIOByte2**
Input	Input Output Other	N/A =('I')	GPIO1CurrStatus =('H'/'L')	GPIO1CurrStatus =('H'/'L')
		N/A =('I')	GPIO1CurrStatus =('H'/'L')	GPIO1CurrStatus =('H'/'L')
		N/A =('I')	GPIO1CurrStatus =('H'/'L')	N/A =('I')
TT	Input Output Other	TTPermStatus =('C'/'O')	TTCurrStatus =('C'/'O')	GPIO1CurrStatus =('H'/'L')
		TTPermStatus =('C'/'O')	TTCurrStatus =('C'/'O')	GPIO1CurrStatus =('H'/'L')
		TTPermStatus =('C'/'O')	TTCurrStatus =('C'/'O')	N/A =('I')
Output	Input Output Other	N/A =('I')	GPIO1CurrStatus =('H'/'L')	GPIO1CurrStatus =('H'/'L')
		N/A =('I')	GPIO1CurrStatus =('H'/'L')	GPIO1CurrStatus =('H'/'L')
		N/A =('I')	GPIO1CurrStatus =('H'/'L')	N/A =('I')
Other	Input Output Other	N/A =('I')	N/A =('I')	GPIO1CurrStatus =('H'/'L')
		N/A =('I')	N/A =('I')	GPIO1CurrStatus =('H'/'L')
		N/A =('I')	N/A =('I')	N/A =('I')

The command is only accepted at application level, and will be rejected at PICC level. Depending on the ReadGPIOAccessCondition, as configured with SetConfiguration Option0x11, the command may require an authentication and specific secure messaging communication mode. By default, the command is disabled.

6.13.3 Mirroring in the NDEF message

The GPIO Input and Tag Tamper statuses can be mirrored together within the NDEF messaging. In this way, the status can be protected by the Secure Dynamic Messaging, as defined in Section 6.3.8 and more specifically Section 6.3.8.5.

If mirroring is enabled, the encoding within the NDEF file will be identical to the 3-byte ReadGPIO response, see Table 255. Note that only Input and Tag Tamper configurations are mirrored. Output configurations are interpreted as 'Other' for NDEF mirroring, i.e. returning 'I' for Invalid.

To enable this mirroring, the configuration must be done with ChangeFileSettings, see Section 7.8.7.

6.13.4 Authentication notification

When configured as output, the GPIO can be configured to notify on authentication. This configuration is also done via SetConfiguration 0x11using the *GPIO1Notif*and*GPIO2Notif*parameter.

When configured, the targeted GPIO is enabled (i.e. set to HIGH representing logical '1') once the authenticated state is reached. This means:

- On successful execution of SIGMA-I mutual authentication; see Section 6.3.2, i.e ISOGeneralAuthenticate replying with message type 0xB4 when acting as responder or 0xA1 when acting as initiator.
- On successful execution of symmetric mutual authentication with AuthenticateEV2NonFirstPart2.

Note that it is still possible to manually toggle the GPIO with ManageGPIO if authentication notification is enabled. Most likely, this kind of double usage of a GPIO pin should be avoided for a use case, as can e.g. be done by configuring the ManageGPIOAccessCondition to 0xF.

When losing the authentication state, the GPIO will be disabled (i.e.set to LOW representing logical '0'). See Section 6.15.1.3 and Section 6.3.3.4 for the different reasons to lose authentication.

6.14 Timer Support

The A30 supports three timers:

- Authority Watchdog Timer 1 (AWDT1)
- Authority Watchdog Timer 2 (AWDT2)
- Halt Watchdog Timer (HWDT)

The timer values are configured using SetConfiguration Option 0x14 as defined in Command SetConfiguration.

6.14.1 Authority Watchdog Timers

The AWDT1 timer is used to limit the time the host has to execute mutual authentication. If configured it is used when performing AES-based symmetric mutual authentication or SIGMA-I asymmetric mutual authentication.

When performing SIGMA-I as the Initiator the AWDT1 timer is started when the A30 sends its ephemeral public key to the host. It is stopped when the host provides its ephemeral public key and session signature or the session is explicitly aborted e.g. a new mutual auth session is started.

When performing AES-based mutual authentication the AWDT1 timer is started when the A30 receives the first AuthenticateEV2First command. It is stopped when the hosts sends the final AuthenticateEV2First command or the session is explicitly aborted e.g. a new mutual auth session is started.

If the AWDT1 timer expires then the A30 closes the current mutual authentication session, and reject the next command received.

The AWDT2 timer is used to limit the period of the secure tunnel opened by a successful mutual authentication. The AWDT2 timer is started when the A30 authenticates the host and completes mutual authentication. When the AWDT2 timer expires the A30 closes the current secure tunnel session.

6.14.2 Halt Watchdog Timer

The HWDT is used to move the A30 to the HALT state to save power when there is no I/O activity. The HWDT timer is only enabled when the device is Vcc powered. The timer is reset (not stopped) when a command is received on the I^2C. resulting in termination of any ongoing activity e.g. an open authentication session.

The HALT state is exited when one of the following events occurs:

- I^2C activity (SDA pulled low).

6.15 ISO/IEC 7816-4 Support

A30 supports ISO/IEC 7816-4 commands [4] by wrapping into ISO/IEC 7816-4 APDUs of the native command set, as explained in Section 6.2.2.

On top, the following standard ISO/IEC 7816-4 commands are supported as well:

- ISOSelectFile with INS code 0xA4
- ISOReadBinary with INS code 0xB0
- ISOUpdateBinary with INS code 0xD6

6.15.1 Standard ISO/IEC 7816-4 commands

A30 supports a selection of standard ISO/IEC 7816-4 commands, i.e. commands from the interindustry class [4]. These commands are defined in this section. First the authentication and secure messaging aspects are of these commands are described.

6.15.1.1 Byte order

For all parameters of standard ISO/IEC 7816-4 commands, the representation on the interface is most significant byte (MSB) first notation. As data like the 2-byte ISO/IEC 7816-4 file identifiers, are in different order on the native command interface, this needs to be especially taken into account.

6.15.1.2 Security concepts of standard ISO/IEC 7816-4 commands

These commands do not support secure messaging, and therefore can only be issued under following conditions:

- ISOReadBinary: if targeted file is configured with at least one of FileAR.Read, FileAR.ReadWrite, FileAR.SDMFileRead to 0xE, i.e. free access, and issuing the command in VCState.NotAuthenticated. Depending on the configuration Secure Dynamic Messaging, see Section 6.3.8, is applied.
- ISOUpdateBinary: if targeted file is configured with at least one of FileAR.Write and FileAR.ReadWrite to 0xE, i.e. free access, and issuing the command in VCState.NotAuthenticated.

6.15.1.3 Error Handling

In case of unsuccessful command execution, A30 sends a return code different from Resp.ISO9000. The full list of ISO/IEC 7816-4 errors is given in Section 7.1.

In case of unsuccessful command execution, A30 executes the same abort actions as for native commands, see Section 6.15.1.3.

The following generic error cases can occur:

- Resp.ISO6985: An ongoing wrapped chained command or multiple pass command is aborted, see Section 6.2.3.
- Resp.ISO6700: Wrong or inconsistent APDU length according to [4].
- Resp.ISO6E00: Unsupported CLA byte.
- Resp.ISO6D00: The received instruction code *INS* is not supported.
- Resp.ISO6A86: Incorrect parameters *P*1 or*P*2.

6.15.1.4 ISOSelectFile

ISOSelectFile as defined in compliance with ISO/IEC7816-4 in Table 257 selects either the PICC level, an application, or a file within the application.

P1 defines the selection method.

If*P*1 is set to 0x00, 0x01, or 0x02, selection is done by a 2-byte ISO file identifier. *P*1 set to 0x00 is used to select the MF (i.e. the PICC level), a DF (i.e. an application if currently the PICC level is selected) or an EF (i.e. a file within the currently selected application). *P*1 set to 0x01 is used to select a DF, if the MF is currently selected.*P*1 set to 0x02 can be used to select an EF, if an application is currently selected. For MF selection, 0x3F00 or empty data is to be used. For DF and EF selection, Data shall hold the 2-byte ISO/IEC 7816-4 file identifier.

Note: *The different byte order for the file identifiers when written with native commands and when used here for ISO/IEC 7816-4 selection.*

If *P*1 is set to 0x03, the MF level is selected. This option can only be issued if currently an application (DF) is selected. In this case, Data must be empty.

If *P*1 is set to 0x04, selection is done by DF name which can be up to 16 bytes.

The registered ISO DF name is 0xD2760000850100. When selecting this DF name, the PICC level (or MF) is selected.

For selecting the application immediately, the ISO/IEC 7816-4 DF name 0xD2760000850101 is to be used.

P2 indicates whether or not File Control Information (FCI) is to be returned in case of application selection.

If this is to be returned, *P*2 is set to 0x00. In this case, FCI is returned as response data, if the following conditions are satisfied:

- the targeted application hold as file with native file number 0x1F.
- this file is of FileType.StandardData.
- this file is freely accessible, i.e. FileAR.Read or FileAR.ReadWrite holds the free access condition, see Section 6.10.2.
- Le is present.

The number of bytes requested by Le up to the complete file data will be returned in plain. There is no specific FCI template format checked, i.e. the data stored in the file will be sent back as is. In case of PICC level or file selection, FCI data is never returned.

In case of failure, the current selection status both at application (MF/DF) and file (EF) level is not affected. The currently selected application and file, if any, remains selected.

6.15.1.5 ISOReadBinary

ISOReadBinary as defined in compliance with ISO/IEC7816-4 in Table 261 can be used to read data from FileType.StandardData files.

P1 and P2 define the targeted file and the offset.

If Bit 7 of *P*1 is set, then *P*1 Bit4-0 encodes a short ISO/IEC 7816-4 file identifier, i.e. referencing the five least significant bits of the 2-byte ISO/IEC 7816-4 file identifiers. All zero bits is reserved for referencing the currently selected file. All one bits is reserved [4] and will be rejected. The referenced file will be selected for this and subsequent operations. Note that if intending to use short file identifiers, the user must take care of avoiding collisions amongst each other and with the reserved values in the definition of the file system, as there is no checking on file creation. *P*2 encodes the offset from 0 byte to 255 byte.

If Bit 7 of *P1* is not set, then *P*1 Bit6-0 concatenated with *P*2 encode the offset from 0 to 32767 byte. The file currently selected is targeted. If no file was selected, the command is rejected. At PICC level, the command is rejected.

Le encodes the number of bytes to be returned. If the encoded value is 0x00 or if it is larger than the number of bytes in the file (starting from the offset), all remaining bytes of the file will be returned.

As listed in Table 37, ISOReadBinary is allowed only if at least one of FileAR.Read, FileAR.ReadWrite and FileAR.SDMFileRead access rights associated with the targeted file is granted. It must be set to 0xE, i.e.free access, as the command is only accepted in VCState.NotAuthenticated, i.e.not supporting EV2 secure messaging.

Only Secure Dynamic Messaging is supported (which does not require a preceding authentication), depending on the targeted file's configuration, see Section 6.3.8.

6.15.1.6 ISOUpdateBinary

ISOUpdateBinary as defined in compliance with ISO/IEC7816-4 in Table 265 can be used to write data to FileType.StandardData files.

P1 and P2 define the targeted file and the offset. The interpretation is identical as for ISOReadBinary, see Section 7.12.3.

At PICC level, the command is rejected.

Lc encodes the number of bytes to be written. The command is rejected if one attempts to write across the file boundary.

The FileType.StandardData does offer limited anti-tearing protection, see Section 6.10.1.1.

As listed in Table 37, ISOUpdateBinary is allowed only if at least one of FileAR.Write and FileAR.ReadWrite access rights associated with the targeted file is granted. It must be set to 0xE, i.e. free access, as the command is only accepted in VCState.NotAuthenticated, i.e. not supporting EV2 secure messaging.

6.16 Trust Provisioning

6.16.1 Originality Check Key Pair and Certificate

During manufacturing, A30 is trust-provisioned with an ECC-based key pair and related certificate to allow verification of the genuineness of the IC. The originality check is done by executing a card-unilateral authentication through a challenge-response protocol. As the protocol creates a trace that potentially cannot be repudiated, the key pair is shared by all ICs in one production batch to reduce the privacy implications. On top, the feature can be disabled through SetConfiguration Option 0x0E.

6.16.1.1 Originality Key Pair

The Originality Check key pair (Priv.Orig, Pub.Orig) is trust-provisioned with the following properties. For KeyPolicy and Read/WriteAccess encoding, refer to ManageKeyPair API definition.

- Shared key pair per batch.
- Priv.Orig is stored as ECCPrivateKey KeyNo 0x01 at the PICC level, with following default configuration:
  - NIST P-256
  - KeyPolicy: 0x0100, only allowing ECC-based Unilateral Authentication.
  - WriteAccess: 0x30. For A30, this access right is irrelevant as A30 does not support reader authentication at the PICC level.
  - KUCLimit: disabled
- Pub.Orig is trusted via the certificate Cert.Orig, as specified in Section 6.16.1.2.

6.16.1.2 Originality Certificate

The Originality Check certificate is stored in a FileType.StandardData file, as introduced in Section 6.10.1.1, at the PICC level.

It can be freely read upfront the authentication using the supported data management (see Section 6.11.1) and standard ISO/IEC 7816-4 commands (see Section 6.15.1) for data file access. Access to the file can be disabled by enabling the enhanced privacy feature, see SetConfiguration.

The file holding the certificate is a FileType.StandardData file of 384 bytes with the following properties:

- FileNo = 0x01; ISO File ID = 0xEF01
- FileAR.Read = 0xE;FileAR.Write = 0x0;FileAR.ReadWrite = 0x0;FileAR.Change = 0x0. For A30, only FileAR.Read is relevant, as one cannot authenticate at the PICC -level.

This file holds the NXP Originality Certificate Cert.Orig. If needed, the file content is further padded with all zero bytes.

The NXP Originality Certificate is signed by NXP Trust Provisioning using a dedicated CA key pair for this product. The CA key pair can be retrieved from https://www.gp-ca.nxp.com/CA/getCA?caid=63709320110003 filling in the CAID with the serialNumber encoded in the issuer name.

The NXP Originality Certificate certificate is a public-key certificate according to X.509 v3 format [29]. Optional fields from [29] have been omitted, that is, the certificate will not contain additional issuerUniqueID, subjectUniqueID, and extensions. The signature algorithm used is ECDSA with SHA-256.

The issuer is set to the following content:

- Organizational name (O, OID 2.5.4.10): “NXP”
- Common name (CN, OID 2.5.4.3): “NXP Orig RootCAvE2xx” with xx varying per product variant
- SerialNumber (OID 2.5.4.5): 14 digits encoding CAID

The subject contains a subset of GetVersion: VendorID || HWMajorVersion || HWMinorVersion || SWType || SWSubType || SWMajorVersion || SWMinorVersion. This is encoded in a description (OID 2.5.4.13)[29], using hexadecimal ASCII encoding.

6.16.1.3 Card-unilateral authentication

The authentication supported by Priv.Orig is outlined in Section 6.3.3.

6.16.2 Application Key Pair and Certificate

At application-level, in the default configuration, A30 is trust-provisioned during manufacturing with an App-level key pair (Priv.App, Pub.App) and related certificate Cert.App. This can be used to authenticate individual devices for executing App-level functionality.

6.16.2.1 Application Key Pair

The application key pair (Priv.App, Pub.App) is a unique key pair per die from which the Priv.App is stored as ECCPrivateKey KeyNo 0x00 within the application, holding with following default configuration:

- NIST P-256
- KeyPolicy: 0x0004, only allowing SIGMA-I Mutual Authentication. Note that by default both Prover and Verifier mode are enabled. For privacy (non-traceability), only Prover may be preferred, with requires configuration per interface (I2C) via SetConfiguration Option 0x0F/0x10.
- WriteAccess: 0x30, allowing replacement with ManageKeyPair after an authentication granting AppMasterKey access rights in CommMode.Full.
- KUCLimit: disabled.

For KeyPolicy and WriteAccess encoding, refer also to ManageKeyPair API definition.

Pub.App is trusted via the certificate Cert.App, as specified in the next subsection.

6.16.2.2 Application Certificate

The Cert.App is stored as an uncompressed end-leaf certificate without parent certificates within a certificate repository, seeSection 6.8.1, with id 0x00, with following default configuration:

- Associated with the ECCPrivateKey with KeyNo 0x00.
- Repository size: [TBD]
- WriteAccess: 0x30, allowing ManageCertRepo after an authentication granting AppMasterKey access rights in CommMode.Full.
- ReadAccess: 0x30, allowing ReadCertRepo after an authentication granting AppMasterKey access rights in CommMode.Full. Note that typically the certificate is only to be retrieved via the SIGMA-I authentication itself. Though if needed this configuration can be changed, requiring the certificate to be re-loaded.
- No mapping table, this means the certificate is stored as a plain X.509 certificate without any PKCS#7 wrapping.

The certificate repository is already activated and thus ready for use with the SIGMA-I authentication.

The Cert.App is signed by NXP Trust Provisioning using a dedicated key pair. The CA key pair can be retrieved from https://www.gp-ca.nxp.com/CA/getCA?caid=63709320101003 filling in the CAID with the serialNumber encoded in the issuer name.

The Cert.App is a public-key certificate according to X.509 v3 format [11]. It has the same structure as the Cert.Orig defined in subsection 17.1.2 with the following content.

The issuer is set to:

- Organizational name (O, OID 2.5.4.10): “NXP”
- Common name (CN, OID 2.5.4.3): “NXP Auth RootCAvE2xx” with xx varying per product variant
- SerialNumber (OID 2.5.4.5): 14 digits encoding CAID

The subject is set to:

- Description (OID 2.5.4.13), containing a subset of GetVersion: VendorID || HWMajorVersion || HWMinorVersion || SWType || SWSubType || SWMajorVersion || SWMinorVersion, using hexadecimal ASCII encoding.
- UniqueIdentifier (OID 2.5.4.45) with 7-byte UID as a BIT STRING

6.16.3 Commercial customization options

NXP provides commercial customization options for trust-provisioning. This allows for a customer-dedicated delivery configuration.

This may include the provisioning of a customer-specific CARootKey. This allows to do the initial personalizaiton with the ECC-based SIGMA-I authentication. By this, the need for a secure environment can be removed, compared to when doing the initial personalization based on the default AES keys.

Additionally, also customer-specific AES keys, certificates and/or a customized file system and configuration can be provisioned. Reach out to your local sales representative for more information.

6.17 Security

6.17.1 Introduction

NXP Semiconductors has gained comprehensive security experience from developing more than six generations of certified secure microcontrollers and other security certified products.

The large number of approved features and the significant enhancements over the different generations of certified secure products and secure microcontrollers are the foundation of the security concept that is implemented in the A30 product. These mentioned design features related to security ensure to protect the integrity and confidentiality of user data and applications.

The unique security design is built on over one hundred dedicated security mechanisms which create a dense protection shield with redundancy and multiple layers. The security mechanisms provide a comprehensive response to the wide variety of known and expected security attacks. As attacks evolve over time, the distributed approach of the implemented security architecture allows for more proactive and continuous enhancements of the security mechanisms compared to alternative and less versatile approaches. This makes the underlying security architecture of A30 a future-proof concept that´s built into the product, that effectively counters side channel and fault attacks as well as reverse engineering efforts.

The following sections describe a subset of the security features that are implemented on A30.

6.17.2 Reset

The following types of resets and reset sources can be distinguished:

- normal application power-on reset, triggered by the on-chip power-on reset circuit
- internal reset, triggerable by
  - Dedicated software reset
  - On-chip security sensors
  - Electrical operating condition category
  - Internal physical attack category
  - Data integrity protection category

Two different reset severities can be distinguished in the A30 hardware. The"normal" chip resets and the "security" resets.

6.17.3 Sensor Architecture

The following sensors are implemented on A30:

- Electrical operation condition category
  - Low Frequency Sensor
  - High Frequency Sensor
  - Low Voltage Sensor
  - High Voltage Sensor
- Internal physical attack category
  - Low Temperature Sensor
  - High Temperature Sensor
  - Light Sensors
  - Glitch Sensors
  - Active Shielding
  - ISO/IEC 14443 Frequency Sensor
- Data integrity protection category
  - RAM Integrity Error
  - ROM Integrity Error
  - FLASH Integrity Error
  - internal Bus and Register Integrity Error

6.17.4 Scalable Security

A30 implements an error counter. The error counter uses a dedicated memory area in the FLASH within a dedicated, protected memory area. The error counter is decremented for security critical errors.

The Scalable Security feature enables additional security countermeasures during operation based on the error counter values to avoid exploitation of repeated attacks. This results in a significant performance degradation in case of repeated security resets, if A30 detects that it is under security attacks. From system design perspective this behavior has to be considered to avoid a non functional system due to timeouts by the host in case of activated Scalable Security features. Therefore timeouts should be defined with significant margins in case of additional security countermeasures are activated.

7 Command set

7.1 Introduction

This section contains the full command set of A30. For each command a figure and a table with the detailed command API is given.

Note: *For non-standard ISO/IEC 7816-4, i.e. proprietary native commands, the command tables show the native command format, i.e. not repeating the CLA/P1/P2/Lc/Le wrapping for each command, while the figures show the wrapped format as supported by A30. For further explanation, see Section 6.2.2.*

Remark: In the figures and tables, always CommMode.Plain is presented and the field length is valid for the plain data length. For the CommMode.MAC and CommMode.Full, the cryptogram needs to be calculated according to the secure messaging, see Section 6.3.6, then data field needs to fill with the cryptogram (Plain; CMAC; encrypted data with CMAC). Communication mode and condition are mentioned in the command description.

7.2 Supported commands and APDUs

Table 42. APDUs

Command	C-APDU (hex)							R-APDU (hex)		Communication mode
INS	CLA	INS	P1	P2	Lc	Data	Le	Data	SW1SW2 Successful	
ActivateConfiguration	90	66	00	00	XX	Data	00	-	9100	CommMode.MAC
AuthenticateEV2First - part 1	90	71	00	00	XX	Data	00	Data	91AF	N/A
AuthenticateEV2First - part 2	90	AF	00	00	20	Data	00	Data	9100	N/A
AuthenticateEV2NonFirst - part 1	90	77	00	00	01	Data	00	Data	91AF	N/A
AuthenticateEV2NonFirst - part 2	90	AF	00	00	20	Data	00	Data	9100	N/A
ChangeFileSettings	90	5F	00	00	XX	Data	00	Data	9100	CommMode.Full
ChangeKey	90	C4	00	00	XX	Data	00	Data	9100	CommMode.Full
CreateCounterFile	90	D0	00	00	8	Data	00	Data	9100	CommMode.MAC
CreateStdDataFile	90	CD	00	00	8	Data	00	Data	9100	CommMode.MAC
CryptoRequest	90	4C	00	00	XX	Data	00	Data	9100	CommMode of Crypto Request as defined by Set Configuration 0x15.
FreeMem	90	6E	00	00	-	-	00	Data	9100	CommMode.MAC
ISOGeneralAuthenticate	00	86	01	00-07	XX	Data	00	Data	9000	N/A
GetCardUID	90	51	00	00	-	-	00	Data	9100	CommMode.Full
GetConfiguration	90	65	00	00	[1]	[Data]	00	Data	9100	CommMode.Full
GetFileIDs	90	6F	00	00	-	-	00	Data	9100	CommMode.MAC
GetISOFileIDs	90	61	00	00	-	-	00	Data	9100	CommMode.MAC
GetFileSettings	90	F5	00	00	1	Data	00	Data	9100	CommMode.MAC
GetFileCounters	90	F6	00	00	1	Data	00	Data	9100	CommMode.Full for SDMReadCtr retrieval on File Type.StandardData; CommMode of targeted file for FileType.Counter
GetKeySettings	90	45	00	00	[1]	[Data]	00	Data	9100	CommMode.MAC

Table 42. APDUs*...continued*

Command	C-APDU (hex)							R-APDU (hex)		Communication mode
INS	CLA	INS	P1	P2	Lc	Data	Le	Data	SW1SW2 Successful	
GetKeyVersion	90	64	00	00	1	Data	00	Data	9100	CommMode.MAC
GetVersion - part 1	90	60	00	00	[1]	[Data]	00	Data	91AF	CommMode.MAC
GetVersion - part 2	90	AF	00	00	-	-	00	Data	91AF	CommMode.MAC
GetVersion - part 3	90	AF	00	00	-	-	00	Data	9100	CommMode.MAC
IncrementCounterFile	90	F8	00	00	5	Data	00	Data	9100	CommMode of targeted file.
ISOInternalAuthenticate	00	88	00	00..04	14..FF	Data	00	Data	9000	N/A
ManageCARootKey	90	48	00	00	XX	Data	00	Data	9100	CommMode of targeted key, or if targeting not yet existing key, default CommMode of the command as defined by SetConfiguration 0x12.
ManageCertRepo	90	49	00	00	XX	Data	00	Data	9100	CommMode of Manage CertRepo as defined by Set Configuration 0x13.
ManageGPIO	90	42	00	00	XX	Data	00	Data	9100	CommMode of Manage GPIO as defined by Set Configuration 0x11.
ManageKeyPair	90	46	00	00	XX	Data	00	Data	9100	CommMode of targeted key, or if targeting not yet existing key, default CommMode of the command as defined by SetConfiguration 0x12.
ProcessSM	90	E5	00	00	XX	Data	00	Data	9100	N/A
ISOReadBinary	00	B0	XX	XX	-	-	00	Data	9000	N/A
ReadCertRepo	90	4A	00	00	2	Data	00	Data	9100	If reading meta-data then CommMode.MAC is applied. Reading a certificate directly from the repository requires access as defined in the Read access condition set during repository creation/ reset.
ReadData	90	AD	00	00	07	Data	00	Data	9100	CommMode of targeted file.
ReadGPIO	90	43	00	00	-	-	00	Data	9100	CommMode of ReadGPIO as defined by SetConfiguration 0x11.
ISOSelectFile	00	A4	04	00	XX	Data	00	Data	9000	N/A
SetConfiguration	90	5C	00	00	XX	Data	00	Data	9100	CommMode.Full
ISOUpdateBinary	00	D6	XX	XX	XX	Data	00	-	9000	N/A
WriteData	90	8D	00	00	XX	Data	00	-	9100	CommMode of targeted file.

7.3 Authentication and Secure Messaging

7.3.1 ISOGeneralAuthenticate

The detailed description of this command can be found in Section 6.3.2.

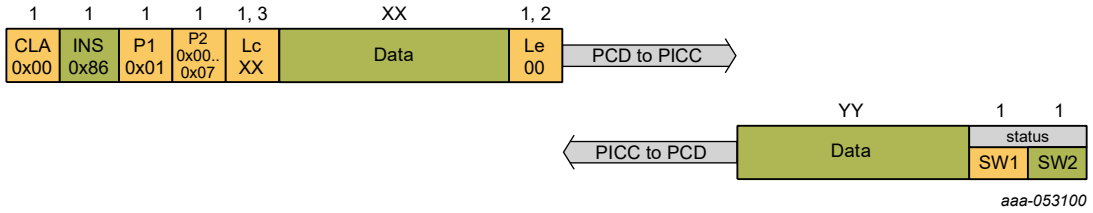


Figure 18. ISOGeneralAuthenticate command protocol

Table 43. Command summary - ISOGeneralAuthenticate

ISOGeneralAuthenticate	
Description:	Asymmetric mutual authentication using SIGMA-I
CommMode:	N/A

Table 44. Command description - ISOGeneralAuthenticate

Name	Length	Value	Description
CLA	1	0x00	
INS	1	0x86	
P1	1	-	Protocol Option
		0x01	SIGMA-I
P2	1	0x00 .. 0x07	Certificate repository Id to use to execute the protocol
Lc	1, 3	0xXX	Length of subsequent data field
Data	XX	-	Message types and payload tags as defined in Table 7 and Table 8
Le	1, 2	0x00	Length of expected response

Table 45. Response description - ISOGeneralAuthenticate

Name	Length	Value	Description
Data	YY	-	Message types and payload tags as defined in Table 7 and Table 8
SW1SW2	2	0x9000 0xXXXX	successful execution Refer to Table 46

Table 46. Error code description - ISOGeneralAuthenticate

SW1|SW2	Value	Description
ISO6E00	0x6E00	Wrong CLA
ISO6A86	0x6A86	Wrong P1 or P2

Table 46. Error code description - ISOGeneralAuthenticate ...continued

SW1|SW2	Value	Description
ISO6700	0x6700	Wrong or inconsistent APDU length.
ISO6985	0x6985	Wrapped chained command or multiple pass command ongoing.
ISO6985	0x6985	Not supported at PICC level.
ISO6985	0x6985	Key usage counter enabled and limit reached
ISO6985	0x6985	Protocol option requested is not supported
ISO6988	0x6988	Invalid host ephemeral public key
ISO6988	0x6988	Host message decryption failed
ISO6A82	0x6A82	Certificate level requested is invalid or certificate has already been requested
ISO6A80	0x6A80	Invalid command data format
ISO6300	0x6300	Verification of host signature failed

7.3.2 ISOInternalAuthenticate

The detailed description of this command can be found in Section 6.3.3.3.

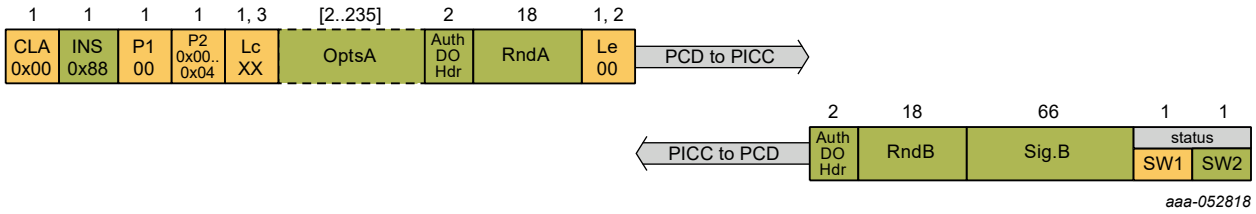


Figure 19. ISOInternalAuthenticate command protocol

Table 47. Command summary - ISOInternalAuthenticate

ISOInternalAuthenticate	
Description:	Asymmetric card-unilateral authentication.
CommMode:	N/A

Table 48. Command Description - ISOInternalAuthenticate

Name	Length	Value	Description
CLA	1	0x00	
INS	1	0x88	
P1	1	0x00	RFU
P2	1	-	Key addressing
	Bit 7-3	'00000'	Reserved
	Bit 2-0	-	RFU
		0x0..0x4	[if PICC level is not selected] At application level, up to five keys are supported.
		0x1	[if PICC level is selected] Priv.Orig

Table 48. Command Description - ISOInternalAuthenticate *...continued*

Name	Length	Value	Description
Lc	1,3	0x14..0xFF	Length of subsequent data field
OptsA	[2..235]	-	PCD Option (TLV): RFU
		T: 0x80	Tag
		L: 0x00..0xE9	Length of Value field. Card will accept other lengths and ignore the Value field.
AuthDOHdr	2	-	Authentication Data Objects Header (TL)
		T: 0x7C	Tag
		L: 0x12	Length of subsequent Authentication Data Objects
RndA	18	-	Authentication Data Object: random challenge from PCD (TLV)
		T: 0x81	Tag
		L: 0x10	Length of Value field
		V: RndA	Value: random challenge
Le	1,2	-	Length of expected response
		0x00/0x0000	Any expected length up to resp. 256/65536 bytes.
		0x56..0xFFFF	Max expected length must be at least 86 bytes.

Table 49. Response description - ISOInternalAuthenticate

Status	Length	Value	Description
AuthDOHdr	2	-	Authentication Data Objects Header (TL)
		T: 0x7C	Tag
		L: 0x54	Length of subsequent Authentication Data Objects
RndB	18	-	Authentication Data Objects: random from PICC (TLV)
		T: 0x7C	Tag
		L: 0x54	Length of subsequent Authentication Data Objects
		V: RndB	Value: random
Sig. B	66	-	Authentication Data Objects: signature from PICC (TLV)
		T: 0x7C	Tag
		L: 0x54	Length of Value field
		V: RndB	Value: $Sig.B = ECDSA_{Sign}(Priv.B; 0xF0F0[\|\|OptsA]\|\|RndB\|\|RndA)$
SW1 || SW2	2	0x9000 0xXXXX	Correct execution Refer to Table 50

Table 50. Error code description - ISOInternalAuthenticate

SW1|SW2	Value	Description
Resp.ISO6700	0x6700	Wrong or inconsistent APDU length.
Resp.ISO6984	0x6984	ECC-based Card-Unilateral Authentication disabled via the key policy of the targeted key.

Table 50. Error code description - ISOInternalAuthenticate *...continued*

SW1|SW2	Value	Description
Resp.ISO6985	0x6985	Originality Check with key 0x1 at PICC level disabled due to enhanced privacy configuration.
Resp.ISO6985	0x6985	Current state different from VCState.NotAuthenticated
Resp.ISO6985	0x6985	ECC-based Card-Unilateral Authentication disabled over I2C interface.
Resp.ISO6985	0x6985	KeyUsageCtrLimit enabled for targeted key has been reached.
Resp.ISO6987	0x6987	Expected DO missing.
Resp.ISO6987	0x6987	Unexpected DO recieved.
Resp.ISO6A86	0x6A86	Wrong parameter P1: different from 0x00.
Resp.ISO6A86	0x6A86	Wrong parameter P2: RFU bits set.
Resp.ISO6A88	0x6A88	Wrong parameter P2: Key targeted by PrivKeyNo does not exist.
Resp.ISO6C00	0x6C00	Wrong Le: expected length insufficient for response data.

7.3.3 AuthenticateEV2First

The detailed description of this command can be found in Section 6.3.4.1.

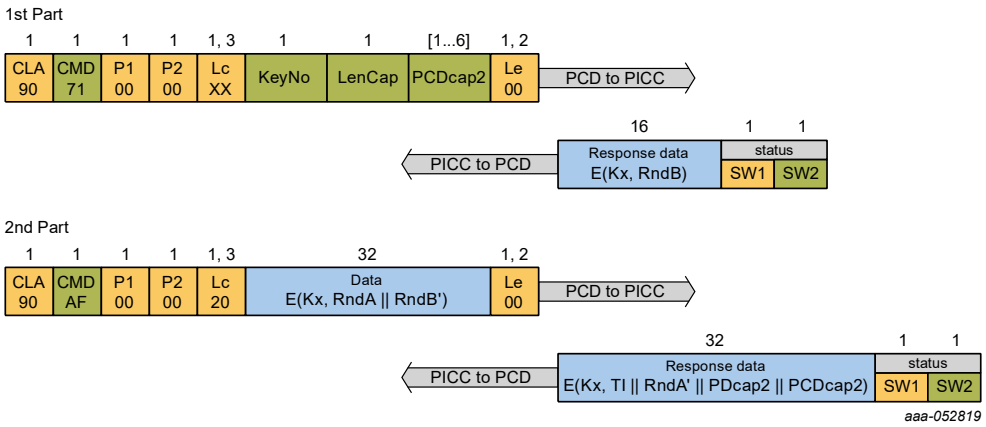


Figure 20. AuthenticateEV2First command protocol

Table 51. Command summary - AuthenticateEV2First

AuthenticateEV2First	
Description:	Symmetric mutual authentication. This authentication is intended to be the first in a transaction.
CommMode:	N/A

Table 52. Command description - AuthenticateEV2First - Part1

Name	Length	Value	Description
Command Header Parameters			
CMD	1	0x71	Command code.
KeyNo	1		Targeted authentication key
	Bit 7-6	00b	RFU
	Bit 5-0	0x0 to 0x4	Key number
LenCap	1	0x00 to 0x06	Length of the PCD Capabilities. [This value should be set to 0x00].
PCDcap2.1	[1]	-	Capability vector of the PCD.
	Bit 7-2	Full range	RFU, can hold any value
	Bit 1	0b	EV2 secure messaging
	Bit 0	Full range	RFU, can hold any value
PCDcap2.2-6	[1..5]	Full range	Capability vector of the PCD. All other bytes but PCDcap2.1 are optional, RFU and can hold any value. [If LenCap set to 0x00, no PCDcap2 present]
Command Data Parameters			
-	-	-	No data parameters

Table 53. Response description - AuthenticateEV2First - Part1

Name	Length	Value	Description
E(Kx, RndB)	16	Full range	Encrypted PICC challenge The following data, encrypted with the key Kx referenced by KeyNo: - RndB: 16 byte random from PICC
SW1SW2	2	0x91AF 0x91XX	successful execution Refer to Table 53

Table 54. Error code description - AuthenticateEV2First - Part1

Status	Value	Description
COMMAND_ABORTED	0xCA	Chained command or multiple pass command ongoing.
LENGTH_ERROR	0x7E	Command size not allowed.
PARAMETER_ERROR	0x9E	Parameter value not allowed.
NO_SUCH_KEY	0x40	Targeted key does not exist
PERMISSION_DENIED	0x9D	Targeted key not available for authentication.
		AES-based Symmetric Authentication disabled over I2C interface.
		AuthCtrLimit enabled for AES-based authentication has been reached.

Table 55. Command description - AuthenticateEV2First - Part2

Name	Length	Value	Description
CMD	1	0xAF	Additional frame
E(Kx, RndA || RndB')	32	Full range	Encrypted PCD challenge and response
			The following data, encrypted with the key Kx referenced by KeyNo: - RndA: 16 byte random from PCD. - RndB': 16 byte RndB rotated left by 1 byte

Table 56. Response description - AuthenticateEV2First - Part2

Name	Length	Value	Description
E(Kx, TI || RndA' || PDcap2 || PCDcap2)	32	Full range	Encrypted PICC response The following data encrypted with the key referenced by KeyNo: - TI: 4 byte Transaction Identifier - RndA': 16 byte RndA rotated left by 1 byte. - PDcap2: 6 byte PD capabilities - PCDcap2: 6 byte PCD capabilities
SW1SW2	2	0x9100 0x91XX	successful execution Refer to Table 57

Table 57. Error code description - AuthenticateEV2First - Part2

Status	Value	Description
COMMAND_ABORTED	0xCA	AWDT1 already expired
LENGTH_ERROR	0x7E	Command size not allowed.
AUTHENTICATION_ERROR	0xAE	Wrong RndB'

7.3.4 AuthenticateEV2NonFirst

The detailed description of this command can be found in Section 6.3.4.2.

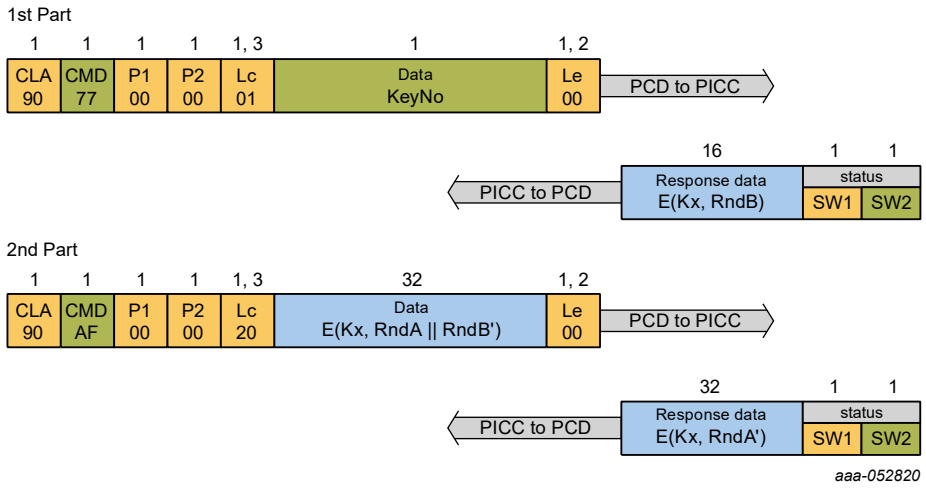


Figure 21. AuthenticateEV2NonFirst command protocol

Table 58. Command summary - AuthenticateEV2NonFirst

AuthenticateEV2NonFirst	
Description:	Symmetric mutual authentication. This authentication is intended for any subsequent authentication after AuthenticateEV2First in a transaction.
CommMode:	N/A

Table 59. Command description - AuthenticateEV2NonFirst - Part1

Name	Length	Value	Description
Command Header Parameters			
CMD	1	0x77	Command code.
KeyNo	1		Targeted authentication key
	Bit 7-6	0	RFU
	Bit 5-0	0x0 to 0x04	Key number
Command Data Parameters			
-	-	-	No data parameters

Table 60. Response description - AuthenticateEV2NonFirst - Part1

Name	Length	Value	Description
E(Kx, RndB)	16	Full range	Encrypted PICC challenge The following data, encrypted with the key Kx referenced by KeyNo: - RndB (16 byte): Random number from the PICC.
SW1SW2	2	0x91AF 0x91XX	successful execution Refer to Table 61

Table 61. Error code description - AuthenticateEV2NonFirst - Part1

Status	Value	Description
COMMAND_ABORTED	0xCA	Chained command or multiple pass command ongoing.
LENGTH_ERROR	0x7E	Command size not allowed.
PARAMETER_ERROR	0x9E	Parameter value not allowed.
NO_SUCH_KEY	0x40	Targeted key does not exist
PERMISSION_DENIED	0x9D	Not in VCState.AuthenticatedAES.
		Targeted key not available for authentication.
		AES-based Symmetric Authentication disabled over I2C interface.

Table 62. Command description - AuthenticateEV2NonFirst - Part2

Name	Length	Value	Description
CMD	1	0xAF	Additional frame
E(Kx, RndA || RndB')	32	Full range	Encrypted PCD challenge and response
			The following data, encrypted with the key Kx referenced by KeyNo: - RndA: 16 byte random from PCD. - RndB': 16 byte RndB rotated left over 1 byte.

Table 63. Response description - AuthenticateEV2NonFirst - Part2

Name	Length	Value	Description
E(Kx, RndA')	16	Full range	Encrypted PICC challenge and response The following data, encrypted with the key Kx referenced by KeyNo: - RndA: 16 byte random from PCD. - RndB': 16 byte RndB rotated left over 1 byte.
SW1SW2	2	0x9100 0x91XX	successful execution Refer to Table 64

Table 64. Error code description - AuthenticateEV2NonFirst - Part2

Status	Value	Description
COMMAND_ABORTED	0xCA	AWDT1 already expired
LENGTH_ERROR	0x7E	Command size not allowed.
AUTHENTICATION_ERROR	0xAE	Wrong RndB'

7.3.5 ProcessSM

The detailed description of this command can be found in subsection Section 6.3.7.1. Instantiations are listed in the subsequent settings. Note that as the regular secure messaging does not apply for these commands, the color coding of the different fields does not apply to distinguish CmdHeader and CmdData parameters.

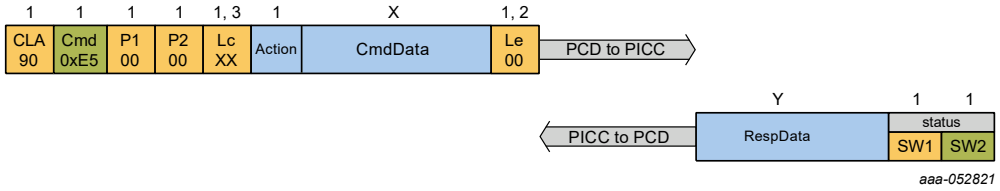


Figure 22. ProcessSM command protocol

Table 65. Command summary - ProcessSM

ProcessSM	
Description:	Processes controller secure messaging. This is the generic API definition, including common error codes. Specific operations are further defined by dedicated subcommands.
CommMode:	N/A

Table 66. Command Description - ProcessSM

Name	Length	Value	Description
Command Header Parameters			
CMD	1	0xE5	Command code.
Command Data Parameters			
Action	1	Full range	Targeted action
CmdData	X	-	Action specific command data

Table 67. Response Description - ProcessSM

Status	Length	Value	Description
RespData	Y	-	Action specific response data
SW1SW2	2	0x9000 0xXXXX	successful execution Refer to Table 68

Table 68. Error code description - ProcessSM

Status	Value	Description
COMMAND_ABORTED	0xCA	Chained command or multiple pass command ongoing.
LENGTH_ERROR	0x7E	Command size not allowed.
PARAMETER_ERROR	0x9E	Parameter value not allowed.
PARAMETER_ERROR	0x9E	Invalid action.
PERMISSION_DENIED	0x9D	ProcessSM disabled for the targeted interface.
PERMISSION_DENIED	0x9D	Not supported at PICC level.

Table 68. Error code description - ProcessSM *...continued*

Status	Value	Description
PERMISSION_DENIED	0x9D	Not supported in VCState.NotAuthenticated
PERMISSION_DENIED	0x9D	Not supported in VCState.AuthenticatedAES

7.3.6 ProcessSM_Apply

This is an instantiation of ProcessSM. The detailed description of this command can be found in Section 6.3.7.2.

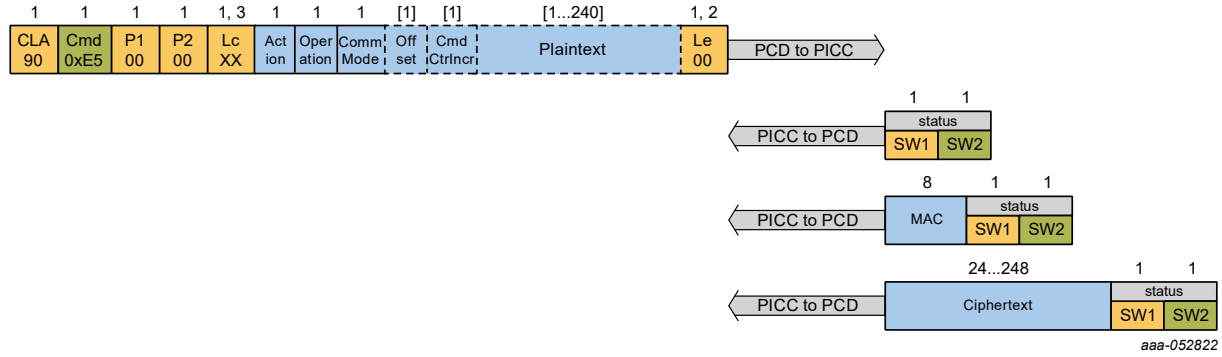


Figure 23. Protocol ProcessSM_Apply

Table 69. Command summary - ProcessSM_Apply

ProcessSM_Apply	
Description:	Applies secure messaging for the given command.
CommMode:	N/A

Table 70. Command Description - ProcessSM_Apply

Name	Length	Value	Description
Command Header Parameters			
CMD	1	0xE5	Command code.
Command Data Parameters			
Action	1	-	Targeted action
		0x01	Apply secure messaging.
Operation	1	-	Targeted action
		0x04	One-shot operation
CommMode	1	-	ProtectionMode
	Bit 7-6	'00'	RFU
	Bit 5-4	-	Communication mode
		'x0'	CommMode.Plain
		'01'	CommMode.MAC
		'11'	CommMode.Full
	Bit 3-0	'0000'	RFU

Table 70. Command Description - ProcessSM_Apply...continued

Name	Length	Value	Description
Offset	[1]	-	[Optional,present if CommMode.Full]
		0x01..0xEF	Index of the first byte of CmdData in Data field.
CmdCtrIncr	[1]	-	[Optional,present if CommMode.Plain]
		0x01..0xFF	Command counter increment value
Plaintext	[1..240]	-	[Optional,present if not CommMode.Plain]
		Full range	Plain data to protect

Table 71. Response Description - ProcessSM_Apply

Status	Length	Value	Description
-	0	-	[if CommMode.Plain] No response data
MAC	8	Full range	[if CommMode.MAC] MAC
Ciphertext	24..248	Full range	[if CommMode.Full] Encrypted data and MAC
SW1SW2	2	0x9000 0xXXXX	successful execution Refer to Table 72

Table 72. Error code description - ProcessSM_Apply

Status	Value	Description
LENGTH_ERROR	0x7E	If CommMode.MAC, Data length bigger than 240 is not supported.
LENGTH_ERROR	0x7E	If CommMode.Full, Data length bigger than 239 is not supported.
INTEGRITY_ERROR	0x1E	If CommMode.Plain,CmdCtr reaches 0xFFFF or overflows.
INTEGRITY_ERROR	0x1E	If CommMode.MAC or CommMode.Full, CmdCtr reached 0xFFFF already.

7.3.7 ProcessSM_Remove

This is an instantiation of ProcessSM. The detailed description of this command can be found in Section 6.3.7.3.

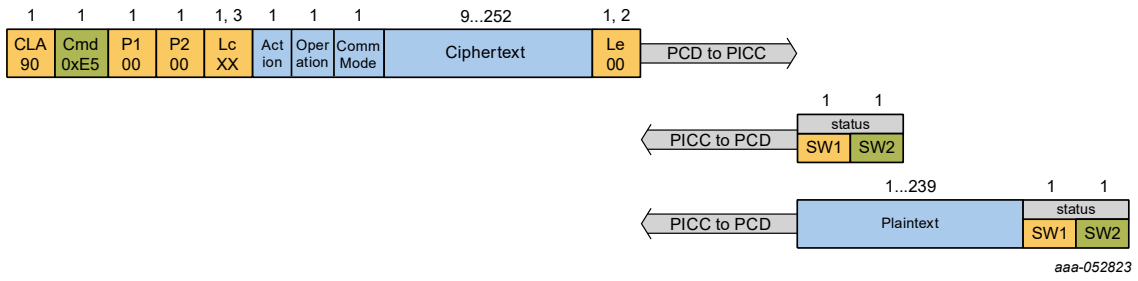


Figure 24. Protocol ProcessSM_Remove

Table 73. Command summary - ProcessSM_Remove

ProcessSM_Remove	
Description:	Applies secure messaging for the given command.
CommMode:	N/A

Table 74. Command Description - ProcessSM_Remove

Name	Length	Value	Description
Command Header Parameters			
CMD	1	0xE5	Command code.
Command Data Parameters			
Action	1	-	Targeted action
		0x02	Remove secure messaging
Operation	1	-	Targeted action
		0x04	One-shot operation
CommMode	1	-	ProtectionMode
	Bit 7-6	'00'	RFU
	Bit 5-4	-	Communication mode
		'x0'	RFU
		'01'	CommMode.MAC
		'11'	CommMode.Full
	Bit 3-0	'0000'	RFU
Ciphertext	9..252	-	Response data
		Full range	[if CommMode.MAC] RC[|| RespData] || MAC
		Full range	[if CommMode.Full] RC[|| encrypted RespData] || MAC
		'01'	CommMode.MAC
		'11'	CommMode.Full

Table 75. Response Description - ProcessSM_Remove

Status	Length	Value	Description
-	0	-	[if CommMode.MAC] No response data
Plaintext	1..239	Full range	[if CommMode.Full] Encrypted data and MAC
SW1SW2	2	0x9000 0xXXXX	successful execution Refer to Table 76

Table 76. Error code description - ProcessSM_Remove

Status	Value	Description
LENGTH_ERROR	0x7E	If CommMode.MAC, Data length bigger than 252 is not supported.
LENGTH_ERROR	0x7E	If CommMode.Full, Data length bigger than 249 is not supported.
INTEGRITY_ERROR	0x1E	Padding error in cryptogram or invalid secure messaging MAC

7.4 Memory and Configuration Management

7.4.1 FreeMem

The detailed description of this command can found in Section 6.5.2.3.1.

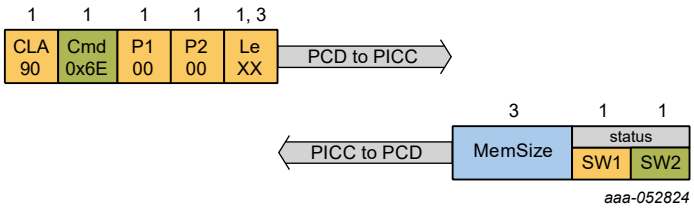


Figure 25. FreeMem command protocol

Table 77. Command summary - FreeMem

FreeMem	
Description:	Returns the free memory available on the card.
CommMode:	CommMode.MAC

Table 78. Command description - FreeMem

Name	Length	Value	Description
Command Header Parameters:			
Cmd	1	0x6E	Command code.
Command Data Parameters:			
-	-	-	No data parameters:

Table 79. Response description - FreeMem - OPERATION_OK

Name	Length	Value	Description
MemSize	3	-	Size of the free memory
SW1SW2	2	0x9100 0x91XX	successful execution Refer to Table 80

Table 80. Error code description - FreeMem

Status	Value	Description
COMMAND_ABORTED	0xCA	Chained command or multiple pass command ongoing.
INTEGRITY_ERROR	0x1E	Invalid secure messaging MAC.
LENGTH_ERROR	0x7E	Command size not allowed.
MEMORY_ERROR	0xEE	Failure when reading or writing to non-volatile memory.

7.4.2 SetConfiguration

The detailed description of this command can found in Command SetConfiguration.

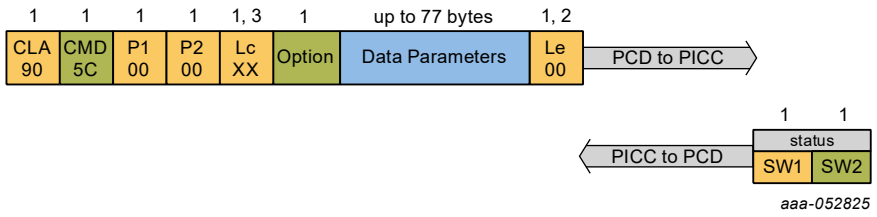


Figure 26. SetConfiguration command protocol

Table 81. Command Description - SetConfiguration

SetConfiguration	
Description:	Configures several aspects of the application.
CommMode:	CommMode.Full

Table 82. Command description - SetConfiguration

Name	Length	Value	Description
Command Header Parameters			
Cmd	1	0x5C	Command code.
Option	1	-	Configuration Option. It defines the length and content of the Data parameter. The Option byte is transmitted in plain text, whereas the Data is always transmitted in CommMode.Full.
		0x00..0x03	RFU
		0x04	Secure Messaging Configuration.
		0x05..0x0F	RFU
		0x10	I^2C Management
		0x11	GPIO Management
		0x12	ECC Key Management
		0x13	Certificate Management
		0x14	Watchdog Timer Management
		0x15	CryptoAPI Management
		0x16	Authentication Counter and Limit Configuration
		0x17	HALT and Wake-up Configuration
		0xFE	Deferred Configurations
		0xFF	Lock Configurations
		Other values	RFU

Table 82. Command description - SetConfiguration *...continued*

Name	Length	Value	Description
Command Data Parameters			
Data	Up to 77 bytes	-	Data content depends on option values.
		Full range	Data content depends on option value as defined in set ConfigOptionsList Table.

Table 83. Response description - SetConfiguration

Name	Length	Value	Description
SW1SW2	2	0x9100 0x91XX	successful execution Refer to Table 84

Table 84. Error code description - SetConfiguration

Status	Value	Description
COMMAND_ABORTED	0xCA	Chained command or multiple pass command ongoing.
INTEGRITY_ERROR	0x1E	Invalid cryptogram (padding or CRC). Invalid secure messaging MAC.
LENGTH_ERROR	0x7E	Command size not allowed. Option 0x00: Data length is not 1 Option 0x02: Data length is not in the range [1..20] Option 0x03: Data length is not 2 Option 0x04: Data length is not 2 Option 0x05: Data length is not 10 Option 0x0C: Data length is not 2 Option 0x0D: Data length is not 1or 3 Option 0x0E: Data length is not 2 Option 0x0F: Data length is not 3 Option 0x10: Data length is not 4 Option 0x11: Data length is not 28 Option 0x12: Data length is not 2 Option 0x13: Data length is not 4 Option 0x14: Data length is not 3 Option 0x15: Data length is not between 3 and 71 Option 0x16: Data length is not 6 Option 0x17: Data length is not 4 Option 0xFE: Unaccepted Data length Option 0xFF: Data length is not 3

Table 84. Error code description - SetConfiguration...continued

Status	Value	Description
PARAMETER_ERROR	0x9E	Parameter value not allowed. Option 0x00: Data bit 7-2 or bit 0 not set to 0b. Option 0x02: TL inconsistent with length of received ATS string. Option 0x02: Data bit 7-2 or bit 0 not set to 0b. Option 0x0D: given REQS equals given WUPS. Option 0x0F: unsupported protocol set. Option 0x10: unsupported protocol set. Option 0x11: unsupported GPIO1Mode, GPIO2Mode, GPIO1Notif, GPIO2 Notif set. Option 0x13: unsupported cache size set. Option 0x13: unsupported feature selected. Option 0x14: unsupported timer value. Option 0x16: unsupported AuthCtrOption. Option 0x17: unsupported configuration. Option 0xFE: unsupported Option or Method values. Unsupported option (i.e. Reserved).
PERMISSION_DENIED	0x9D	Option not supported / allowed at PICC level Option not supported by product configuration
FILE_NOT_FOUND	0xF0	Option 0x16: invalid AuthCtrFileID: file does not exist.
AUTHENTICATION_ERROR	0xAE	No active authentication with AppMasterKey.
CERT_ERROR	0xAE	Active ECC-based authentication not granting AppMasterKey access rights.

7.4.3 GetConfiguration

The detailed description of this command can be found in Section 6.5.2.2.

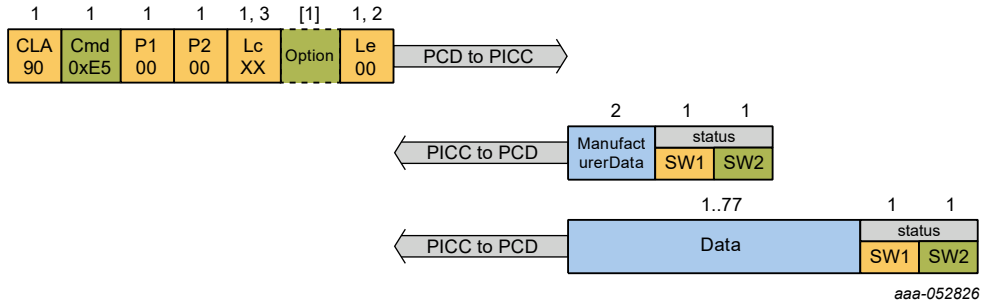


Figure 27. GetConfiguration command protocol

Table 85. Command summary - GetConfiguration

GetConfiguration	
Description:	Retrieves configuration aspects of the card or the application.
CommMode:	CommMode.Full

Table 86. Command Description - GetConfiguration

Name	Length	Value	Description
Command Header Parameters			
CMD	1	0x65	Command code.
Option	[1]	-	Configuration Option. If absent, manufacturer configuration data is returned.
		Limited range	For supported options, see SetConfiguration.

Table 87. Response description - GetConfiguration

Status	Length	Value	Description
ManufacturerData	2	-	[if no Option provided]
Data	1..77	-	[if Option provided] Data content and length depends on option value as defined in Table 1.
SW1SW2	2	0x9100 0x91XX	successful execution Refer to Table 88

Table 88. Error code description - GetConfiguration

Status	Value	Description
COMMAND_ABORTED	0xCA	Chained command or multiple pass command ongoing.
INTEGRITY_ERROR	0x1E	Invalid secure messaging MAC.
LENGTH_ERROR	0x7E	Command size not allowed.
PARAMETER_ERROR	0x9E	Unsupported Option.
PERMISSION_DENIED	0x9D	Option not supported at PICC level.
PERMISSION_DENIED	0x9D	Option disabled by product configuration, see SetConfiguration.
AUTHENTICATION_ERROR	0xAE	No active authentication with required key for the issued Option, see SetConfiguration.
AUTHENTICATION_ERROR	0xAE	No active authentication with AppMasterKey if issued without Option.
CERT_ERROR	0xCE	Active ECC-based authentication not granting access rights for the issued Option, see SetConfiguration.
CERT_ERROR	0xCE	Active ECC-based authentication not granting AppMasterKey access rights if issued without Option.

7.4.4 ActivateConfiguration

The detailed description of this command can be found in Section 7.4.4.

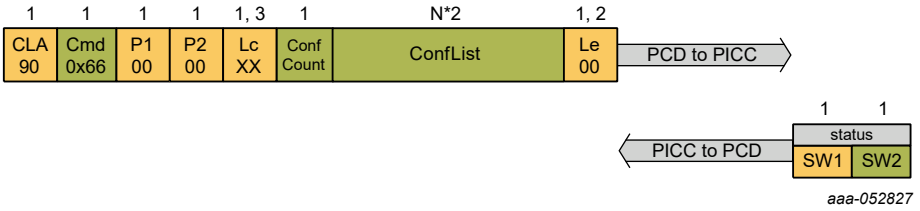


Figure 28. ActivateConfiguration command protocol

Table 89. Command summary - ActivateConfiguration

ActivateConfiguration	
Description:	Activates a deferred configuration.
CommMode:	CommMode.MAC

Table 90. Command Description - ActivateConfiguration

Name	Length	Value	Description
Command Header Parameters			
CMD	1	0x66	Command code.
ConfCount	1	0x01 .. 0x04	Number of configurations to be activated (N).
ConfList	N*2	-	List of configurations to be activated (with size N*2). List must hold one or more of following values.
		0x5C 0x00	activate SetConfiguration 0x01 (RandomID)
		0x5C 0x0D	activate SetConfiguration 0x0D (Silent Mode)
		0x5C 0x11	activate SetConfiguration 0x11 (TagTamper boot measurements)
		0x5F 0x01	activate ChangeFileSettings SDM encryptions
Command Data Parameters			
-	-	-	No data parameters

Table 91. Response description - ActivateConfiguration

Status	Length	Value	Description
ManufacturerData	2	-	[if no Option provided]
Data	1..77	-	[if Option provided] Data content and length depends on option value as defined in Table 1.
SW1SW2	2	0x9100 0x91XX	successful execution Refer to Table 91

Table 92. Error code description - ActivateConfiguration

Status	Value	Description
OPERATION_OK	0x00	
COMMAND_ABORTED	0xCA	Chained command or multiple pass command ongoing.
INTEGRITY_ERROR	0x1E	Invalid secure messaging MAC.
LENGTH_ERROR	0x7E	Command size not allowed.
PARAMETER_ERROR	0x9E	Parameter value not allowed.
PERMISSION_DENIED	0x9D	Parameter value not configured for ActivateConfiguration or already activated.

7.4.5 GetVersion

The detailed description of this command can be found in Section 6.5.1.1.

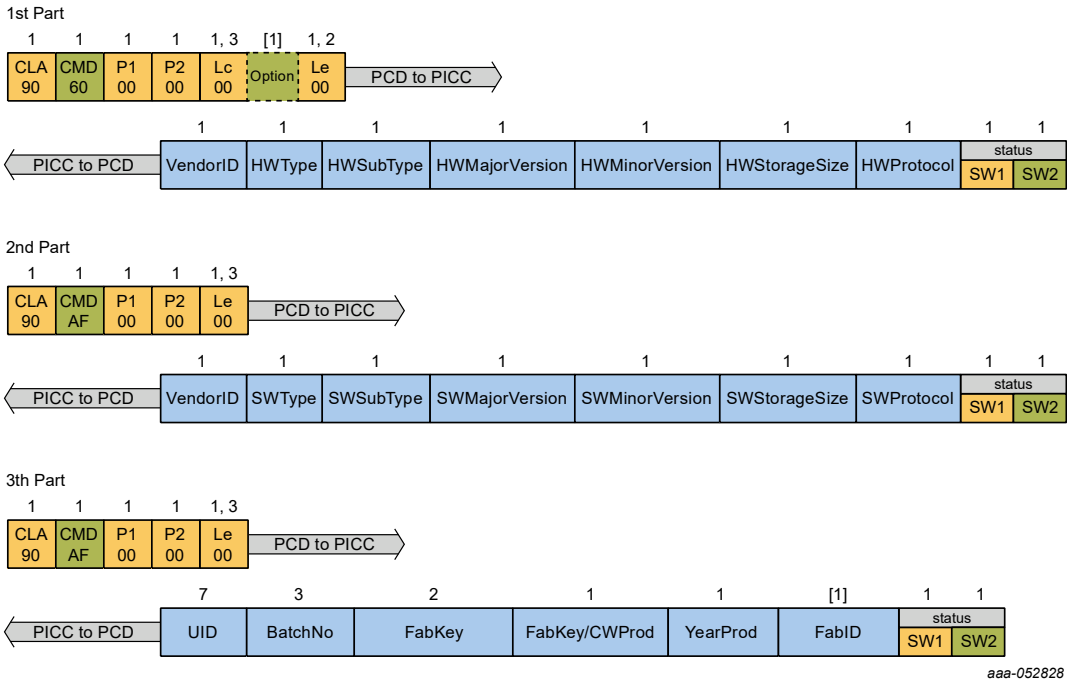


Figure 29. GetVersion command protocol

Table 93. Command summary - GetVersion

GetVersion	
Description:	Returns manufacturing related data.
CommMode:	CommMode.MAC

Part 1

Table 94. Command parameters description - GetVersion - Part1

Name	Length	Value	Description
Command Header Parameters			
Cmd	1	0x60	Command code.

Table 94. Command parameters description - GetVersion - Part1 *...continued*

Name	Length	Value	Description
Option	[1]	-	[Optional] Option byte
		0x01	Return Fab Identifier
Command Data Parameters			
-	-	-	No data parameters

Table 95. Response description - GetVersion - Part1

Name	Length	Value	Description
VendorID	1	0x04	Vendor ID
HWType	1	0x0A	HW type for IoT
HWSubType	1	-	HW subtype
		0x41	17 pF, Tag Tamper
		0x43	50 pF, Tag Tamper
HWMajorVersion	1	0xA0	HW major version number
HWMinorVersion	1	0x00	HW minor version number
HWStorageSize	1	-	HW storage size
		0x1A	8 KB
		0x1C	16 KB
		other values	RFU
HWProtocol	1	-	HW communication protocol type
		0x15	ISO/IEC 14443-4 support with Silent Mode support
		0x20	I^2C
		0x35	I^2C and ISO/IEC 14443-4 support with Silent Mode support
SW1SW2	2	0x91AF 0x91XX	successful execution Refer to Table 100

Part 2

Table 96. Command parameters description - GetVersion - Part2

Name	Length	Value	Description
CMD	1	0xAF	Additional frame request.
Data	0	-	No data parameters:

Table 97. Response description - GetVersion - Part2

Name	Length	Value	Description
VendorID	1	0x04	Vendor ID
SWType	1	0x0A	SW type for IoT
SWSubType	1	0x01	SW subtype

Table 97. Response description - GetVersion - Part2 *...continued*

Name	Length	Value	Description
SWMajorVersion	1	0x00	SW major version number
SWMinorVersion	1	0x01	SW minor version number
SWStorageSize	1	-	SW storage size
		0x1A	8 KB
		0x1C	16 KB
		other values	RFU
SWProtocol	1	-	SW communication protocol type
		0x15	ISO/IEC 14443-4 support with Silent Mode support
		0x20	I^2C
		0x35	I^2C and ISO/IEC 14443-4 support with Silent Mode support
SW1SW2	2	0x91AF 0x91XX	successful execution Refer to Table 100

Part 3

Table 98. Command parameters description - GetVersion - Part3

Name	Length	Value	Description
CMD	1	0xAF	Additional frame request.
Data	0	-	No data parameters:

Table 99. Response description - GetVersion - Part3

Name	Length	Value	Description
UID	7	-	UID
		All zero	if configured for RandomID
		Full range	UID if not configured for RandomID
BatchNo	3	-	Production batch number
		All zero	if manufacturer data masking is enabled
FabKeyID	2	-	
		Limited range	AlphaNumeric ASCII encoding
		All zero	if manufacturer data masking is enabled
CWProd	1	-	Calendar week of production
		0x01..0x52	BCD coding
		All zero	if manufacturer data masking is enabled
YearProd	1	-	Year of production
		Full range	if manufacturer data masking is disabled
		All zero	if manufacturer data masking is enabled

Table 99. Response description - GetVersion - Part3 ...continued

Name	Length	Value	Description
FabID	[1]	-	[Optional, present if Option = 0x01] Fab Identifier
		Full range	FabID mapping
		All zero	if manufacturer data masking is enabled
SW1SW2	2	0x9100 0x91XX	Successful execution Refer to Table 100

Table 100. Error code description - GetVersion

Status	Value	Description
COMMAND_ABORTED	0xCA	Chained command or multiple pass command ongoing.
INTEGRITY_ERROR	0x1E	Invalid secure messaging MAC (only).
LENGTH_ERROR	0x7E	Command size not allowed.
PARAMETER_ERROR	0x9E	Parameter value not allowed.

7.4.6 GetCardUID

The detailed description of this command can be found in Command.

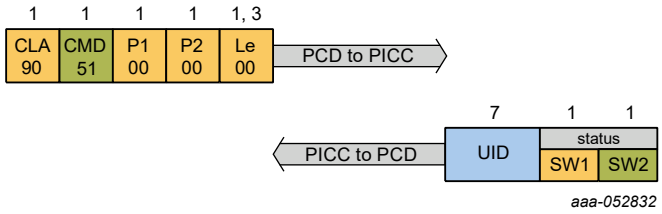


Figure 30. GetCardUID command protocol

Table 101. Command summary - GetCardUID

GetCardUID	
Description:	Returns manufacturing related data.
CommMode:	CommMode.Full

Table 102. Command parameters description - GetCardUID

Name	Length	Value	Description
Command Header Parameters			
Cmd	1	0x51	Command code.
Command Data Parameters			
-	-	-	No data parameters

Table 103. Response description - GetCardUID

Name	Length	Value	Description
UID	7	Full range	UID of the A30

Table 103. Response description - GetCardUID ...continued

Name	Length	Value	Description
SW1SW2	2	0x9100 0x91XX	successful execution Refer to Table 104

Table 104. Error code description - GetCardUID

Status	Value	Description
COMMAND_ABORTED	0xCA	Chained command or multiple pass command ongoing.
INTEGRITY_ERROR	0x1E	Invalid secure messaging MAC (only).
LENGTH_ERROR	0x7E	Command size not allowed.
PERMISSION_DENIED	0x9D	Not supported at PICC level.
AUTHENTICATION_ERROR	0xAE	No active authentication
AUTHENTICATION_ERROR	0xAE	No active authentication with AppPrivacyKey while enabled.
CERT_ERROR	0xCE	Active ECC-based authentication not granting AppPrivacyKey access rights while enabled.

7.5 Symmetric Key management

7.5.1 ChangeKey

The detailed description of this command can be found in Section 6.6.4.1.

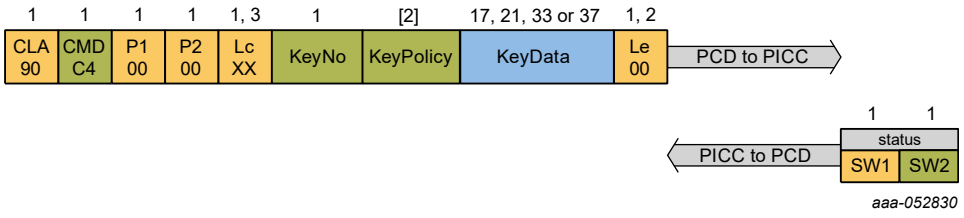


Figure 31. ChangeKey command protocol

Table 105. Command summary - ChangeKey

ChangeKey	
Description:	This command updates a symmetric key.
CommMode:	CommMode.Full

Table 106. Command description - ChangeKey

Name	Length	Value	Description
Command Header Parameters			
Cmd	1	0xC4	Command code.

Table 106. Command description - ChangeKey *...continued*

Name	Length	Value	Description
KeyNo	1	-	Key number of the key to be changed.
	Bit 7-6	-	[if targeting AppMasterKey or CryptoRequestKey] Key type
		10b	KeyType.AES128
		11b	KeyType.AES256
		00b	[else] RFU
	Bit 5-0		Key Number
		0x0..0x4	AppMasterKey
		0x10..0x17	CryptoRequestKey
KeyPolicy	[2]	-	[Optional, present if targeting CryptoRequestKey] Defines the allowed crypto operations with the targeted key.
	Bit 15-9	'0'	RFU
	Bit 8	-	HKDF
		'0'	disabled
		'1'	enabled
	Bit 7	-	HMAC
		'0'	disabled
		'1'	enabled
	Bit 6	-	GCM/CCM Encrypt/Sign with internal NONCE only
		'0'	disabled
		'1'	enabled
	Bit 5	-	GCM/CCM Encrypt/Sign
		'0'	disabled
		'1'	enabled
	Bit 4	-	GCM/CCM Decrypt/Verify
		'0'	disabled
		'1'	enabled
	Bit 3	-	ECB/CBC Encrypt
		'0'	disabled
		'1'	enabled
	Bit 2	-	ECB/CBC Decrypt
		'0'	disabled
		'1'	enabled
	Bit 1	-	MAC Sign
		'0'	disabled
		'1'	enabled

Table 106. Command description - ChangeKey *...continued*

Name	Length	Value	Description
	Bit 0	-	MAC Verify
		'0'	disabled
		'1'	enabled
Command Data Parameters			
KeyData	17, 21, 33, 37		New key data.
		full range (17/33-byte length)	[targeting CryptoRequestKey or AppMasterKey] NewKey || KeyVer
		full range (21/37-byte length)	[targeting AppKey different from AppMasterKey] (NewKey XOR OldKey) || KeyVer || CRC32NK [1]

[1] The CRC32NK is the 4-byte CRC value computed according to IEEE Std 802.3-2008 (FCS Field) over NewKey [10]

Table 107. Response description - ChangeKey

Name	Length	Value	Description
SW1SW2	2	0x9100 0x91XX	successful execution Refer to Table 108

Table 108. Error code description - ChangeKey

Status	Value	Description
COMMAND_ABORTED	0xCA	Chained command or multiple pass command ongoing.
INTEGRITY_ERROR	0x1E	Integrity error in cryptogram or invalid secure messaging MAC (Secure Messaging).
LENGTH_ERROR	0x7E	Command size not allowed.
PARAMETER_ERROR	0x9E	Parameter value not allowed.
NO_SUCH_KEY	0x40	Targeted key does not exist
PERMISSION_DENIED	0x9D	Not allowed at PICC level.
PERMISSION_DENIED	0x9D	Not allowed to set both HMAC-related (bit 8-7) and AES-related (bit 6-0) bits in the KeyPolicy.
AUTHENTICATION_ERROR	0xAE	At application level, missing active authentication with AppMasterKey while targeting any AppKey.
AUTHENTICATION_ERROR	0xAE	At application level, missing active authentication granting Set Configuration Option 0x15 ChangeAC access rights while targeting any CryptoRequestKey.
CERT_ERROR	0xCE	Active ECC-based authentication not granting AppMasterKey while targeting any AppKey.
CERT_ERROR	0xCE	Active ECC-based authentication not granting SetConfiguration Option 0x15 ChangeAC access rights while targeting any Crypto RequestKey.

7.5.2 GetKeySettings

The detailed description of this command can found in Section 6.6.4.2.

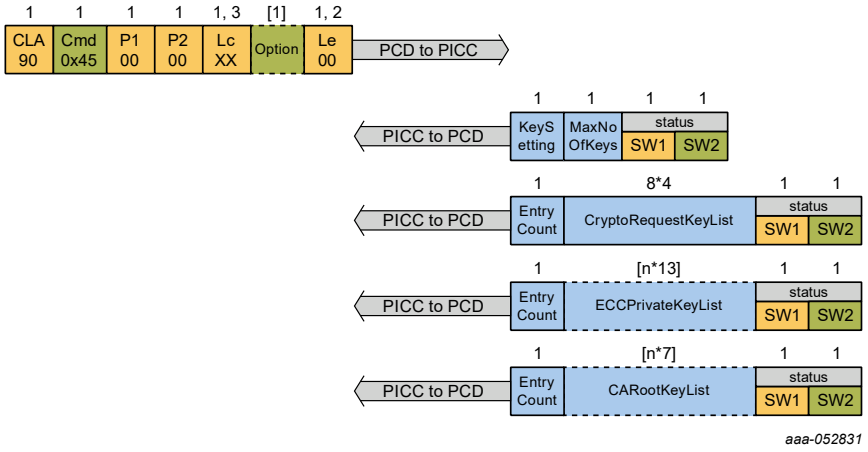


Figure 32. GetKeySettings command protocol

Table 109. Command Description - GetKeySettings

GetKeySettings	
Description:	This command retrieves the meta-data of certain key types.
CommMode:	CommMode.MAC

Table 110. Command description - GetKeySettings

Name	Length	Value	Description
Command Header Parameters:			
Cmd	1	0x45	Command code.
Option	[1]	-	
		0x00	CryptoRequestKeys meta-data
		0x01	ECCPrivateKey meta-data
		0x02	CARootKeys meta-data
Command Data Parameters:			
-	-	-	No data parameters:

Table 111. Response description - GetKeySettings - [No Option byte provided]

Name	Length	Value	Description
KeySetting	1	0x03	Reserved
MaxNoOfKeys	1	-	Maximum number of keys which can be stored within the selected application. Additionally the key type is returned.
	Bit 7-6		Key type
		00b	Reserved
		01b	Reserved
		10b	KeyType.AES128
		11b	KeyType.AES256
	Bit 5-0		Number of keys
		0x05	Number of application keys.
SW1SW2	2	0x9100 0x91XX	successful execution Refer to Table 115.

Table 112. Response description - GetKeySettings - [Option = 0x00] CryptoRequestKey's meta-data

Name	Length	Value	Description
EntryCount	1	0x08	Number of key information entries (n) that will follow.
CryptoRequestKey List	8 × 4	-	List with meta-data
		Entry[0]	KeyNo
		Entry[1]	KeyType: KeyType.AES128 or KeyType.AES256.
		Entry[2..3]	KeyPolicy, see ChangeKey.
SW1SW2	2	0x9100 0x91XX	successful execution Refer to Table 115.

Table 113. Response description - GetKeySettings - [Option = 0x01] ECCPrivateKey's meta-data

Name	Length	Value	Description
EntryCount	1	0x00..0x05	Number of key information entries (n) that will follow.
ECCPrivateKeyList	[n*13]	-	List with meta-data, see ManageKeyPair.
		Entry[0]	KeyNo
		Entry[1]	CurveID
		Entry[2..3]	KeyPolicy
		Entry[4]	WriteAccess
		Entry[5..8]	KeyUsageCtrLimit
		Entry[9..12]	KeyUsageCtr
SW1SW2	2	0x9100 0x91XX	successful execution Refer to Table 115.

Table 114. Response description - GetKeySettings - [Option = 0x02] CARootKey's meta-data

Name	Length	Value	Description
EntryCount	1	0x00..0x05	Number of key information entries (n) that will follow.
CARootKeyList	[n*7]	-	List with meta-data, see ManageCARootKey.
		Entry[0]	KeyNo
		Entry[1]	CurveID
		Entry[2..3]	AccessRights
		Entry[4]	WriteAccess
		Entry[5]	Reserved
		Entry[6]	Reserved
SW1SW2	2	0x9100 0x91XX	successful execution Refer to Table 115.

Table 115. Error code description - GetKeySettings

Status	Value	Description
COMMAND_ABORTED	0xCA	Chained command or multiple pass command ongoing.
INTEGRITY_ERROR	0x1E	Invalid secure messaging MAC
LENGTH_ERROR	0x7E	Command size not allowed.
PARAMETER_ERROR	0x9E	Parameter value not allowed.
PERMISSION_DENIED	0x9D	Option different from 0x01 not supported at PICC level.
AUTHENTICATION_ERROR	0xAE	No active authentication with AppMasterKey.
CERT_ERROR	0xCE	Active ECC-based authentication not granting AppMaster Key access rights.

7.5.3 GetKeyVersion

The detailed description of this command can found in Section 6.6.4.3.

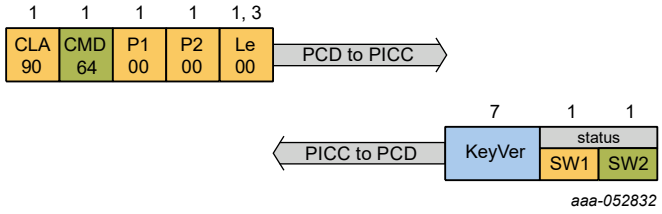


Figure 33. GetKeyVersion command protocol

Table 116. Command Description - GetKeyVersion

GetKeyVersion	
Description:	This command retrieves the key version of the key targeted.
CommMode:	CommMode.MAC

Table 117. Command parameters description - GetKeyVersion

Name	Length	Value	Description
Command Header Parameters			
Cmd	1	0x64	Command code.
KeyNo	1	-	Key number of the targeted key
	Bit 7-6	'00'	RFU
	Bit 5-0	0x0..0x4	AppKey
		0x10..0x17	CryptoRequestKey
Command Data Parameters			
-	-	-	No data parameters

Table 118. Response description - GetKeyVersion

Name	Length	Value	Description
KeyVer	1	Full range	Key version of the targeted key
SW1SW2	2	0x9100 0x91XX	successful execution Refer to Table 119

Table 119. Error code description - GetKeyVersion

Status	Value	Description
COMMAND_ABORTED	0xCA	Chained command or multiple pass command ongoing.
INTEGRITY_ERROR	0x1E	Invalid secure messaging MAC (only).
LENGTH_ERROR	0x7E	Command size not allowed.
PARAMETER_ERROR	0x9E	Parameter value not allowed.

Table 119. Error code description - GetKeyVersion ...continued

Status	Value	Description
PERMISSION_DENIED	0x9D	Not supported at PICC level
NO_SUCH_KEY	0x40	Targeted key does not exist.

7.6 Asymmetric Key Management

7.6.1 ManageKeyPair

The detailed description of this command can be found in Section 6.7.1.1.

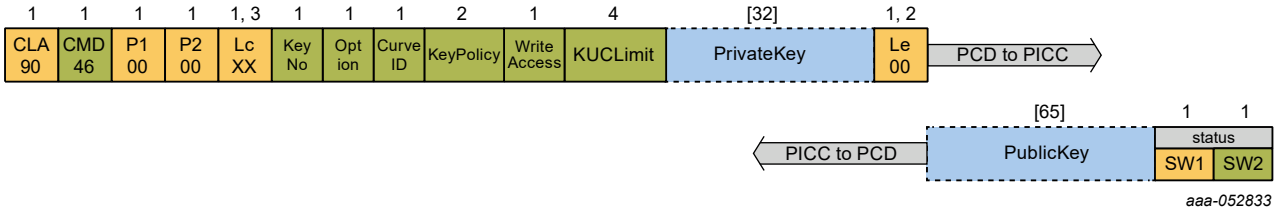


Figure 34. ManageKeyPair command protocol

Table 120. ManageKeyPair

ManageKeyPair	
Description:	Creates or updates a private key entry by generating a key pair or importing a private key.
CommMode:	CommMode of targeted key, or if targeting not yet existing key, default CommMode of the command as defined by SetConfiguration 0x12.

Table 121. Command Description - ManageKeyPair

Name	Length	Value	Description
Command Header Parameters			
CMD	1	0x46	Command code.
KeyNo	1	-	Key number of the key to be managed.
	Bit 7-3	'00000'	RFU
	Bit 2-0	-	KeyNo
		0x0..0x4	Up to five keys are supported
Option	1	-	Targeted action
		0x00	Generate Key Pair
		0x01	Import Private Key
		0x02	Update metadata
CurveID	1	-	Targeted curve
		0x0C	NIST P-256
		0x0D	brainpoolP256r1

Table 121. Command Description - ManageKeyPair *...continued*

Name	Length	Value	Description
KeyPolicy	2	-	Defines the allowed crypto operations with the targeted key.
	Bit 15	-	Freeze KeyUsageCtrLimit
		'0'	disabled
		'1'	enabled
	Bit 14- 9	'000000'	RFU
	Bit 8	-	ECC-based Card-Unilateral Authentication with ISOInternal Authenticate
		'0'	disabled
		'1'	enabled
	Bit 7-6	'00'	Reserved
	Bit 5	-	ECC-based Secure Dynamic Messaging
		'0'	disabled
		'1'	enabled
	Bit 4	-	CryptoRequest ECC Sign (Action 0x03)
		'0'	disabled
		'1'	enabled
	Bit 3	-	CryptoRequest ECC DH (Action 0x05)
		'0'	disabled
		'1'	enabled
	Bit 2	-	SIGMA-I Mutual Authentication
		'0'	disabled
		'1'	enabled
	Bit 1-0	'00'	Reserved
WriteAccess	1	-	Defines the CommMode and access right required to update the key with ManageKeyPair
	Bit 7-6	'00'	RFU
	Bit 5-4	-	Write CommMode (see Table 14)
		'x0'	CommMode.Plain
		'01'	CommMode.MAC
		'11'	CommMode.Full
	Bit 3-0	-	WriteAR
		Full range	Access condition (see Table 17)
KUCLimit	4	-	Defines the key usage limit of the targeted key.
		0x00000000	KeyUsageCtrLimit disabled
		0x00000001 .. 0xFFFFFFFF	KeyUsageCtrLimit enabled with the given value (LSB first).

Table 121. Command Description - ManageKeyPair *...continued*

Name	Length	Value	Description
Command Data Parameters			
PrivateKey	[32]	Full range	[Optional, present if Option is set to 0x01] Private key to be imported

Table 122. Response description - ManageKeyPair

Name	Length	Value	Description
PublicKey -	[65]	Limited range	[Optional, present if Option is set to 0x00] Uncompressed public key: `0x04||Pub:x||Pub:y`
SW1SW2	2	0x9100 0x91XX	successful execution Refer to Table 123

Table 123. Error code description - ManageKeyPair

Status	Value	Description
COMMAND_ABORTED	0xCA	Chained command or multiple pass command ongoing.
INTEGRITY_ERROR	0x1E	Integrity error in cryptogram or invalid secure messaging MAC
LENGTH_ERROR	0x7E	Command size not allowed.
PARAMETER_ERROR	0x9E	Parameter value not allowed.
NO_SUCH_KEY	0x40	Targeting nonexisting key while trying to update metadata.
PERMISSION_DENIED	0x9D	Not allowed at PICC level.
PERMISSION_DENIED	0x9D	Targeting nonexisting key while ManageKeyPair access condition is set to 0xF.
PERMISSION_DENIED	0x9D	Targeting existing key with its WriteAccess condition set to 0xF.
PERMISSION_DENIED	0x9D	Trying to update metadata, setting CurveID to a value different from the curve associated with the targeted key.
PERMISSION_DENIED	0x9D	Trying to enable key for both ECDH (KeyPolicy Bit 3) and ECDSA (other Key Policy Bits) operations.
PERMISSION_DENIED	0x9D	Trying to reconfigure a frozen KeyUsageCtrLimit via updating metadata by setting Key Policy Bit 15 to '0' or KUCLimit to a different value than the currently configured KeyUsageCtrLimit.
PERMISSION_DENIED	0x9D	Trying to import key while disabled by product configuration.
PERMISSION_DENIED	0x9D	Trying to update metadata while disabled by product configuration.
AUTHENTICATION_ ERROR	0xAE	Targeting nonexisting key while ManageKeyPair access condition is not granted while different from 0xF
AUTHENTICATION_ ERROR	0xAE	Targeting existing key while the WriteAccess condition of the targeted key different from 0xF not being granted.
CERT_ERROR	0xCE	Targeting nonexisting key with an active ECC-based authentication while ManageKeyPair access condition is not granted while different to 0xF.
CERT_ERROR	0xCE	Targeting existing key with an active ECC-based authentication while the WriteAccess condition of the targeted key different from 0xF not being granted.

Table 123. Error code description - ManageKeyPair *...continued*

Status	Value	Description
OUT_OF_MEMORY_ ERROR	0x0E	Insufficient free user memory available for creating new key.

7.6.2 ManageCARootKey

The detailed description of this command can be found in Section 6.7.2.1.

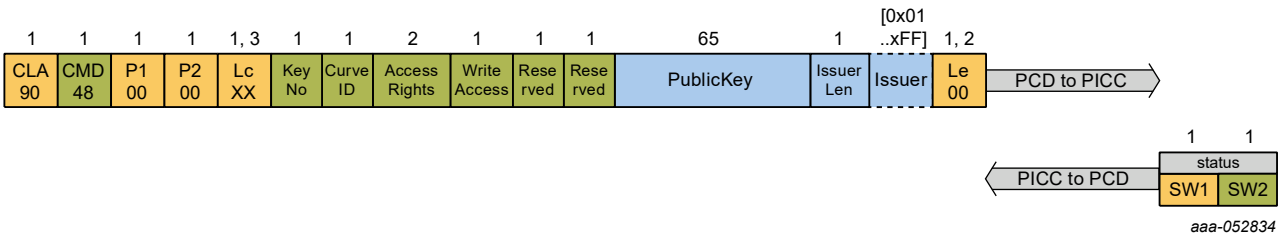


Figure 35. ManageCARootKey command protocol

Table 124. ManageCARootKey

ManageCARootKey	
Description:	Creates or updates a public key entry.
CommMode	CommMode of targeted key, or if targeting not yet existing key, default CommMode of the command as defined by SetConfiguration 0x12.

Table 125. Command Description - ManageCARootKey

Name	Length	Value	Description
Command Header Parameters			
Cmd	1	0x48	Command code.
KeyNo	1	-	Key number of the key to be managed.
	Bit 7-3	'00000'	RFU
	Bit 2-0	-	KeyNo
		0x0..0x4	Up to five keys are supported
CurveID	1	-	Targeted curve
		0x0C	NIST P-256
		0x0D	brainpoolP256r1
AccessRights	2	Limited range	Access rights associated with the CARootKey, see Table 18.
WriteAccess	1	-	Defines the CommMode and access rights required to update the key with ManageCARootKey
	Bit 7-6	'00'	RFU
	Bit 5-4	-	Write CommMode (see Table 14).
		'x0'	CommMode.Plain
		'01'	CommMode.MAC
		'11'	CommMode.Full

Table 125. Command Description - ManageCARootKey *...continued*

Name	Length	Value	Description
	Bit 3-0	-	WriteAR
		Full range	Access condition (see Table 17).
Reserved	1	0x3F	
Reserved	1	0x00	
Command Data Parameters:			
PublicKey	65	-	CA Public Key
		Limited range	Uncompressed public key: `0x04||Pub:x||Pub:y`
IssuerLen	1	-	Length of trusted issuer name
		0x00	No trusted issuer name check required.
		0x01..0xFF	Length of Issuer.
Issuer	[0x01.. xFF]	Full range	[Optional, present if IssuerLen != 0x00] Trusted issuer name of `IssuerLen` bytes.

Table 126. Response description - ManageCARootKey

Name	Length	Value	Description
SW1SW2	2	0x9100 0x91XX	successful execution Refer to Table 127

Table 127. Error code description - ManageKeyPair

Status	Value	Description
Resp.COMMAND_ABORTED	0xCA	Chained command or multiple pass command ongoing.
Resp.INTEGRITY_ERROR	0x1E	Integrity error in cryptogram or invalid secure messaging MAC
Resp.LENGTH_ERROR	0x7E	Command size not allowed.
Resp.PARAMETER_ERROR	0x9E	Parameter value not allowed.
Resp.PERMISSION_DENIED	0x9D	Not allowed at PICC level.
Resp.PERMISSION_DENIED	0x9D	Targeting nonexisting key while ManageCARootKey access condition is set to 0xF.
Resp.PERMISSION_DENIED	0x9D	Targeting existing key with its WriteAccess condition set to 0xF.
Resp.AUTHENTICATION_ERROR	0xAE	Targeting nonexisting key while ManageCARootKey access condition is not granted while different from 0xF.
Resp.AUTHENTICATION_ERROR	0xAE	Targeting existing key while the WriteAccess condition of the targeted key different from 0xF not being granted.
Resp.CERT_ERROR	0xCE	Targeting nonexisting key with an active ECC-based authentication while ManageCARootKey access condition is not granted while different to 0xF.
Resp.CERT_ERROR	0xCE	Targeting existing key with an active ECC-based authentication while the WriteAccess condition of the targeted key different from 0xF not being granted.

Table 127. Error code description - ManageKeyPair *...continued*

Status	Value	Description
Resp.OUT_OF_MEMORY_ERROR	0x0E	Insufficient free user memory available for creating this file.

7.6.3 GetKeySettings

See Section 7.5.2.

7.7 Certificate Management

7.7.1 ManageCertRepo

The detailed description of this command's usage can be found in Section 6.8.1.

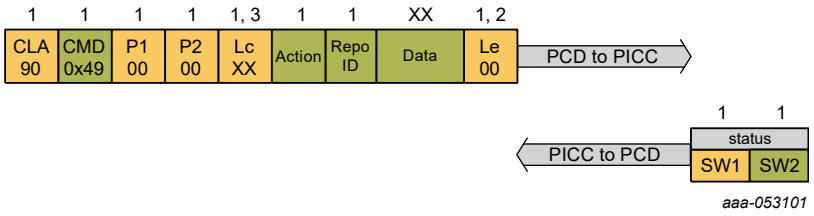


Figure 36. ManageCertRepo command protocol

ManageCertRepo	
Description:	Manages Certificate Repositories
CommMode:	CommMode of ManageCertRepo as defined by SetConfiguration 0x13.

Table 128. Command Description - ManageCertRepo

Name	Length	Value	Description
Command Header Parameters			
CMD	1	0x49	Command code.
Command Action	1	-	The first byte of the command data specifies the action
		0x00	Create certificate repository
		0x01	Load certificate
		0x02	Load Certificate Mapping Information
		0x04	Activate Repository
		0x05	Reset Certificate Repository
Certificate Repository Id	1	0x00 - 0x07	ID used to identify certificate repository for algorithm execution and repository modification. ***Note:*** *The certificate Id shall be used to reference a private key/certificate chain when performing SIGMA-I*

Table 128. Command Description - ManageCertRepo *...continued*

Name	Length	Value	Description
Remaining Data	5	Table 129	[if option is create certificate repository]
	6 - 691	Table 130	[if option is load certificate]
	1 - 650	Table 131	[if option is load certificate mapping info]
	0	-	[if option is activate certificate repository]
	2	Table 132	[if option is reset certificate repository]

Table 129. ManageCertRepo - Create Certificate Repository

Name	Length	Value	Description
Private Key Id	1	0x00 - 0x04	ID of ECC private key associated with this repository (key must have been created using ManageKey Pair).
Repository Size	2	0x0001 - 0x1400	Number of bytes of NVM memory to reserve for the certificate repository
Certificate Repository Write/ Reset Access	1	-	Defines the access right required to write or reset this repository using the ManageCertRepo command
		Bit 7-6	RFU
		Bit 5-4	CommMode, see Table 14.
		Bit 3-0	AccessCondition Value, see Table 17.
Certificate Repository Read Command Access	1	-	Defines the access right required to read from this repository using the ReadCertRepo command
		Bit 7-6	RFU
		Bit 5-4	CommMode, see Table 14.
		Bit 3-0	AccessCondition Value, see Table 17.

Table 130. ManageCertRepo - Load Certificate

Name	Length	Value	Description
Certificate Level	1	0x00	End-leaf
		0x01	Parent
		0x02	Grand-parent
Certificate	3 - 654	-	Certificate Data Bytes. The maximum length of certificate is 650 bytes. Either: 0x7f21, length, uncompressed cert Or 0x7f22, length, compressed cert 0x99, 0x20, cert hash (only for end-leaf cert)

Table 131. ManageCertRepo - Load Certificate Mapping info

Name	Length	Value	Description
Certificate Level	1	0x00	End-leaf
		0x01	Parent
		0x02	Grand-parent
Certificate Mapping Data Length	2	0x0001 - 0x028A	
Certificate Mapping Data	1-650	-	Mapping Data

Table 132. ManageCertRepo - Reset Certificate Repository

Name	Length	Value	Description
Certificate Repository Write/ Reset Access	1	-	Defines the access right required to write or reset this repository using the ManageCertRepo command (actions 0x01 to 0x05)
		Bit 7-6	RFU
		Bit 5-4	CommMode, see Table 14.
		Bit 3-0	AccessCondition Value, see Table 17.
Certificate Repository Read Command Access	1	-	Defines the access right required to read from this repository using the ReadCertRepo command
		Bit 7-6	RFU
		Bit 5-4	CommMode, see Table 14.
		Bit 3-0	AccessCondition Value, see Table 17.

Table 133. ManageCertRepo - Error Conditions

Status	Value	Description
Resp.COMMAND_ABORTED	0xCA	Chained command or multiple pass command ongoing.
Resp.INTEGRITY_ERROR	0x1E	MAC does not match data.
Resp.LENGTH_ERROR	0x7E	Command size not allowed. No MAC provided. Padding bytes wrong length
Resp.PARAMETER_ERROR	0x9E	Parameter value not allowed.
Resp.PERMISSION_DENIED	0x9D	Not supported at PICC level.
Resp.PERMISSION_DENIED	0x9D	Access Condition is 0xF
Resp.PERMISSION_DENIED	0x9D	Attempt to write to an activated certificate repository
Resp.PERMISSION_DENIED	0x9D	Attempt to activate a certificate repository which does not contain a leaf certificate
Resp.AUTHENTICATION_ ERROR	0xAE	No active authentication granting the Access Condition while different from 0x0F
Resp.BOUNDARY_ERROR	0xBE	Attempt to write data to beyond certificate repository limits
Resp.CERT_ERROR	0xCE	Active ECC-based authentication while Access Condition not granted while different from 0xF

Table 133. ManageCertRepo - Error Conditions...*continued*

Status	Value	Description
Resp.DUPLICATE_ERROR	0xDE	Attempt to certificate repository, which already exists
Resp.FILE_NOT_FOUND	0xF0	Certificate repository specified does not exist
Resp.NO_SUCH_KEY	0x40	Private key specified does not exist

7.7.2 ReadCertRepo

The detailed description of this command's usage can be found in Section 6.8.1.

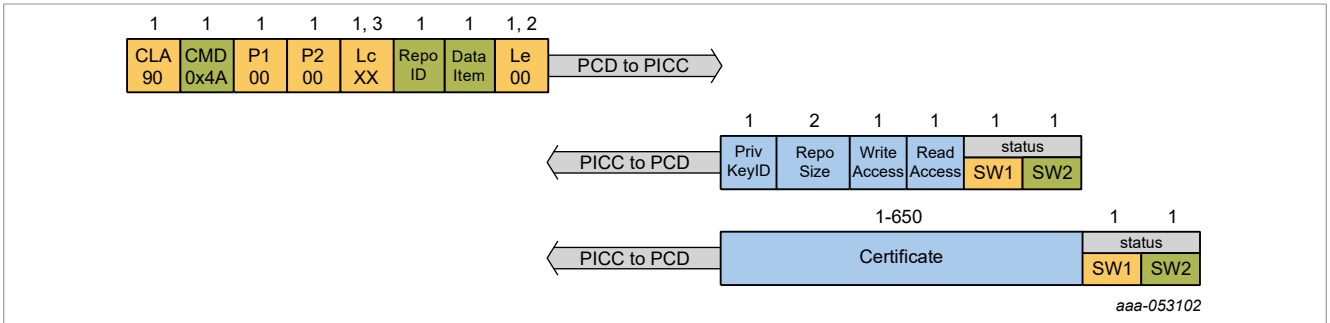


Figure 37. ReadCertRepo command protocol

ReadCertRepo	
Description:	Returns information related to certificate repositories
CommMode:	If reading metadata, then CommMode.MAC is applied. Reading a certificate directly from the repository requires access as defined in the Read access condition set during repository creation/reset.

Table 134. Command Description - ReadCertRepo

Name	Length	Value	Description
Command Header Parameters			
CMD	1	0x4A	Command code.
Certificate Repository Id	1	0x00 - 0x07	Id used to identify certificate repository
Data Item	1	0x00	End-leaf
		0x01	Parent
		0x02	Grand-parent
		0xFF	Repository metadata

Table 135. ReadCertRepo - Response Data Format for Metadata

Name	Length	Value	Description
Private Key Id	1	0x00 - 0x04	Id of ECC private key associated with this repository (key must have been created using ManageKeyPair).
Repository Size	2	0x01 - 0x1400	Number of bytes of NVM memory to reserve for the certificate repository

Table 135. ReadCertRepo - Response Data Format for Metadata *...continued*

Name	Length	Value	Description
Certificate Repository Write/ Reset Access	1		Access required to write or reset this repository using the ManageCertRepo command
		Bit 7-6	RFU
		Bit 5-4	CommMode, see Table 14.
		Bit 3-0	AccessCondition Value, see Table 17
Certificate Repository Read Command Access	1		Access required to read from this repository using the Read CertRepo command
		Bit 7-6	RFU
		Bit 5-4	CommMode, see Table 14.
		Bit 3-0	AccessCondition Value, see Table 17.

Table 136. ReadCertRepo - Response Data Format for Certificate

Name	Length	Value	Description
Certificate	1 - 650	-	Certificate Data Bytes

Table 137. Error Code Description - ReadCertRepo

Status	Value	Description
Resp.COMMAND_ABORTED	0xCA	Chained command or multiple pass command ongoing.
Resp.INTEGRITY_ERROR	0x1E	MAC does not match data.
Resp.LENGTH_ERROR	0x7E	Command size not allowed. No MAC provided. Padding bytes wrong length
Resp.PARAMETER_ERROR	0x9E	Parameter value not allowed.
Resp.PERMISSION_DENIED	0x9D	Not supported at PICC level.
Resp.PERMISSION_DENIED	0x9D	Access condition is 0xF
Resp.AUTHENTICATION_ERROR	0xAE	No active authentication granting the Access Condition while different from 0xF and not requesting metadata
Resp.CERT_ERROR	0xCE	Active ECC-based authentication while Access Condition not granted while different from 0xF and not requesting metadata
Resp.FILE_NOT_FOUND	0xF0	Certificate repository specified does not exist
Resp.CERT_NOT_FOUND	0xC0	Certificate does not exist in the certificate repository

7.8 File Management

7.8.1 CreateStdDataFile

The detailed description of this command can be found in Section 6.10.4.1.

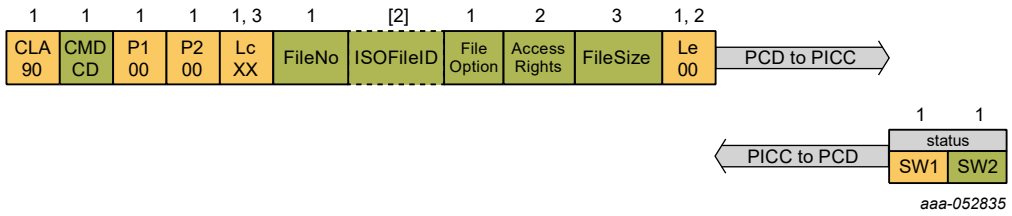


Figure 38. CreateStdDataFile command protocol

Table 138. Command Description - CreateStdDataFile

CreateStdDataFile	
Description:	Creates files for the storage of plain unformatted user data.
CommMode:	CommMode.MAC

Table 139. Command description - CreateStdDataFile

Name	Length	Value	Description
Command Header Parameters:			
Cmd	1	0xCD	Command code.
FileNo	1	-	File number of the file to be created.
	Bit 7		Second Application Indicator
		0b	Target primary application
		1b	Target secondary application
	Bit 6-5		RFU
	Bit 4-0		File number
ISOFileID	[2]	-	[Optional] ISO/IEC 7816-4 File ID for the file to be created.
		Full Range	Excluding the following values reserved by ISO: 0x0000 0x3F00, 0x3FFF, 0xFFFF.
FileOption	1	-	Options for the targeted file.
	Bit 7		Additional Access Rights
		0b	disabled
		1b	enabled
	Bit 6	-	Secure Dynamic Messaging and Mirroring
		0b	disabled
		1b	enabled
	Bit 5-2	0000b	RFU

Table 139. Command description - CreateStdDataFile...*continued*

Name	Length	Value	Description
	Bit 1-0	-	CommMode (see Table CommunicationModes)
		x0b	CommMode.Plain
		01b	CommMode.MAC
		11b	CommMode.Full
AccessRights	2	-	Set of access conditions for the 1st set in the file (see Setaccessconditions Table).
FileSize	3	-	File size in bytes for the file to be created.
		0x00001 .. 0xFF FFFF	Empty file not allowed.
Command Data Parameters:			
-	-	-	No data parameters

Table 140. Response description - CreateStdDataFile

Name	Length	Value	Description
SW1SW2	2	0x9100 0x91XX	successful execution Refer to Table 141

Table 141. Error code description - CreateStdDataFile

Status	Value	Description
COMMAND_ABORTED	0xCA	Chained command or multiple pass command ongoing.
INTEGRITY_ERROR	0x1E	Invalid secure messaging MAC.
LENGTH_ERROR	0x7E	Command size not allowed.
		ISO/IEC 7816-4 File ID is enabled for the targeted application but not present in the received command.
PARAMETER_ERROR	0x9E	Parameter value not allowed.
		Targeted key for one of the access conditions in CreateStd DataFile.AccessRights does not exist.
PERMISSION_DENIED	0x9D	Not supported at PICC level.
		SAI given but no 2nd application selected.
		Trying to pre-enable SDM on File 0x1F.
		Trying to pre-enable SDM while the application is not of Key- Type.AES.
AUTHENTICATION_ERROR	0xAE	No active authentication with AppMasterKey while required by AppKeySettings.
DUPLICATE_ERROR	0xDE	File with the targeted CreateStdDataFile.FileNo or Create StdDataFile.ISOFileID already exists.

Table 141. Error code description - CreateStdDataFile...*continued*

Status	Value	Description
OUT_OF_MEMORY_ERROR	0x0E	Conventional application: insufficient free user memory available for creating this file.
		Delegated application: QuotaLimit of targeted delegated application exceeded if creating this file.
MEMORY_ERROR	0xEE	Failure when reading or writing to nonvolatile memory.

7.8.2 CreateCounterFile

The detailed description of this command can be found in Section 6.10.4.2.

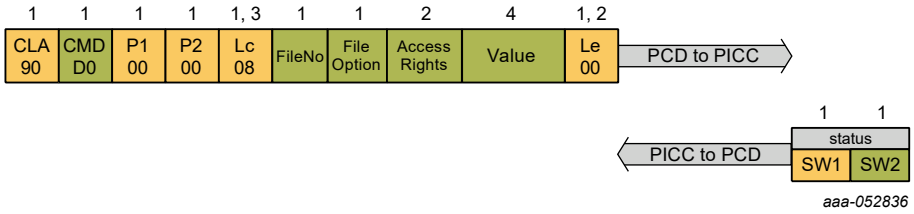


Figure 39. CreateCounterFile command protocol

Table 142. CreateCounterFile

CreateCounterFile	
Description:	Creates a Counter File.
CommMode:	CommMode.MAC

Table 143. Command Description - CreateCounterFile

Name	Length	Value	Description
Command Header Parameters			
CMD	1	0xD0	Command code.
FileNo	1	-	File number of the file to be created.
	Bit 7	'0'	Reserved
	Bit 6-5	'00'	RFU
	Bit 4-0	Full Range	File number
FileOption	1	-	Options for the targeted file
	Bit 7-2	'000000'	RFU
	Bit 1-0	-	CommMode (see Table 14)
		'X0'	CommMode.Plain
		'01'	CommMode.MAC
		'11'	CommMode.Full
AccessRights	2	Limited range	Set of access conditions (see Table 17).
Value	4	Full Range	Current Value

Table 144. Response description - CreateCounterFile

Name	Length	Value	Description
SW1SW2	2	0x9100 0x91XX	successful execution Refer to Table 145

Table 145. Error code description - CreateCounterFile

Status	Value	Description
Resp.COMMAND_ABORTED	0xCA	Chained command or multiple pass command ongoing.
Resp.INTEGRITY_ERROR	0x1E	Invalid secure messaging MAC.
Resp.LENGTH_ERROR	0x7E	Command size not allowed.
Resp.PARAMETER_ERROR	0x9E	Parameter value not allowed.
Resp.PERMISSION_DENIED	0x9D	Parameter value not configured for ActivateConfiguration or already activated.
Resp.PERMISSION_DENIED	0x9D	Trying to create FileType.Counter while disabled by product configuration.
Resp.AUTHENTICATION_ERROR	0xAE	No active authentication with AppMasterKey.
Resp.CERT_ERROR	0xCE	Active ECC-based authentication not granting AppMasterKey access rights.
Resp.DUPLICATE_ERROR	0xDE	File with the targeted FileNo already exists.
Resp.OUT_OF_MEMORY_ERROR	0x0E	Insufficient free user memory available for creating this file.

7.8.3 GetFileIDs

The detailed description of this command can be found in Section 6.10.3.3.

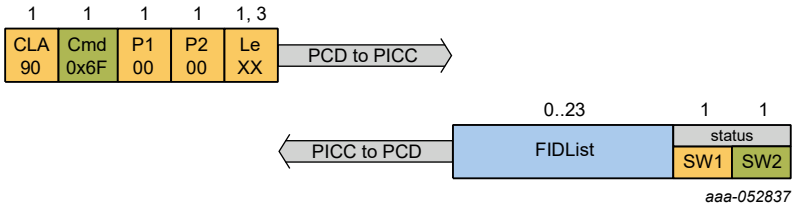


Figure 40. GetFileIDs command protocol

Table 146. Command Description - GetFileIDs

GetFileIDs	
Description:	Returns the File IDentifiers of all active files within the currently selected application.
CommMode:	CommMode.MAC

Table 147. Command description - GetFileIDs

Name	Length	Value	Description
Command Header Parameters:			
Cmd	1	0x6F	Command code.

Table 147. Command description - GetFileIDs*...continued*

Name	Length	Value	Description
Command Data Parameters:			
-	-	-	No data parameters

Table 148. Response description - GetFileIDs

Name	Length	Value	Description
FIDList	0..32	-	List of n File IDs
SW1SW2	2	0x9100 0x91XX	successful execution Refer to Table 149

Table 149. Error code description - GetFileIDs

Status	Value	Description
COMMAND_ABORTED	0xCA	Chained command or multiple pass command ongoing.
INTEGRITY_ERROR	0x1E	Invalid secure messaging MAC.
LENGTH_ERROR	0x7E	Command size not allowed.
PARAMETER_ERROR	0x9E	Parameter value not allowed.
PERMISSION_DENIED	0x9D	Not supported at PICC level.
AUTHENTICATION_ERROR	0xAE	No active authentication with AppMasterKey while required from AppKeySettings.
MEMORY_ERROR	0xEE	Failure when reading or writing to nonvolatile memory.

7.8.4 GetISOFileIDs

The detailed description of this command can be found in Section 6.10.3.4.

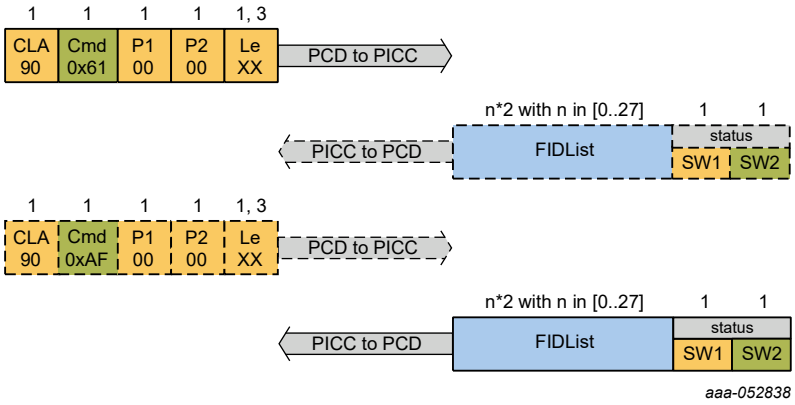


Figure 41. GetISOFileIDs command protocol

Table 150. Command Description - GetISOFileIDs

GetISOFileIDs	
Description:	Get back the ISO File IDs.
CommMode:	CommMode.MAC

Table 151. Command description - GetISOFileIDs

Name	Length	Value	Description
Command Header Parameters:			
Cmd	1	0x61	Command code.
Command Data Parameters:			
-	-	-	No data parameters:

Table 152. Response description - GetISOFileIDs

Name	Length	Value	Description
FIDList	n*2 with n in [0..27]	-	List of n ISO File IDs.
SW1SW2	2	0x91AF	successful execution - more data expected. Command chaining is only applied if the list does not fit into one response frame. In this case, the list is split between two ISO File IDs, i.e. never a partial ISO File ID is sent.
		0x9100	successful execution
		0x91XX	Refer to Table 153

Table 153. Error code description - GetISOFileIDs

Status	Value	Description
COMMAND_ABORTED	0xCA	Chained command or multiple pass command ongoing.
INTEGRITY_ERROR	0x1E	Invalid secure messaging MAC.
LENGTH_ERROR	0x7E	Command size not allowed.
PARAMETER_ERROR	0x9E	Parameter value not allowed.
PERMISSION_DENIED	0x9D	Not supported at PICC level.
FILE_NOT_FOUND	0xF0	Application was created with ISO/IEC 7816-4 file identifiers disabled.
AUTHENTICATION_ERROR	0xAE	No active authentication with AppMasterKey while required from AppKeySettings.
MEMORY_ERROR	0xEE	Failure when reading or writing to nonvolatile memory.

7.8.5 GetFileSettings

The detailed description of this command can be found in Section 6.10.3.1.

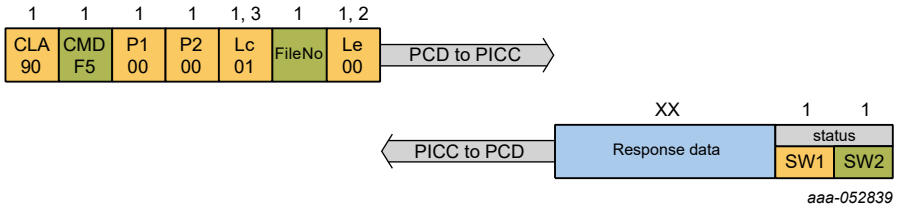


Figure 42. GetFileSettings command protocol

Table 154. Command Description - GetFileSettings

GetFileSettings	
Description:	Get information on the properties of a specific file.
CommMode:	CommMode.MAC

Table 155. Command description - GetFileSettings

Name	Length	Value	Description
Command Header Parameters			
Cmd	1	0xF5	Command code.
FileNo	1	-	File number of the targeted file.
	Bit 7-5		RFU
	Bit 4-0		File number
Command Data Parameters			
-	-	-	No data parameters

Table 156. Response description - GetFileSettings - Targeting FileType.StandardData

Name	Length	Value	Description
FileType	1	-	File Type of the targeted file.
		0x00	StandardData File
		Other values	RFU
FileOption	1	-	Options for the targeted file.
	Bit 7		RFU
	Bit 6	-	Secure Dynamic Messaging and Mirroring
		0b	disabled
		1b	enabled
	Bit 5-4	00b	RFU
	Bit 3	-	Deferred Configuration
		0b	disabled
		1b	enabled
	Bit 2	0b	RFU
	Bit 1-0		CommMode (see Table 14)
AccessRights	2	-	Set of access conditions for the 1st set in the file (see Section 6.10.2).
FileSize	3	-	File size of the targeted file.
SDMOptions	[1]	-	[Optional, present if FileOption[Bit 6] set] SDM Options, see Table 165
SDMAccessRights	[2]	-	[Optional, present if FileOption[Bit 6] set] SDM Access Rights, see Table 165
UIDOffset	[3]	-	[Optional, present if ((SDMOptions[Bit 7] = 1b) AND (SDMMeta Read access right = 0xE)] Mirror position (LSB first) for UID, see Table 165
SDMReadCtrOffset	[3]	-	[Optional, present if ((SDMOptions[Bit 6] = 1b) AND (SDMMeta Read access right = 0xE)] Mirror position (LSB first) for SDMReadCtr, see Table 165
PICCDataOffset	[3]	-	[Optional, present if SDMMetaRead access right =0x0..0x4] Mirror position (LSB first) for encrypted PICCData, see Table 165
GPIOStatusOffset	[3]	-	[Optional, present if (SDMOptions[Bit 3] = 1b)] Mirror position (LSB first) for GPIOStatus, see Table 165
SDMMACInputOffset	[3]	-	[Optional, present if SDMFileRead access right != 0xF] Offset in the file where the SDM MAC computation starts (LSB first), see Table 165
SDMENCOffset	[3]	-	[Optional, present if ((SDMFileRead access right != 0xF) AND (SDMOptions[Bit 4] = 1b))] SDMENCFileData mirror position (LSB first), see Table 165

Table 156. Response description - GetFileSettings - Targeting FileType.StandardData *...continued*

Name	Length	Value	Description
SDMENCLength	[3]	-	[Optional, present if ((SDMFileRead access right != 0xF) AND (SDMOptions[Bit 4] = 1b))] Length of the SDMENCFileData (LSB first), see Table 165
SDMMACOffset	[3]	-	[Optional, present if SDMFileRead access right != 0xF] SDMMAC mirror position (LSB first), see Table 165
SDMReadCtrLimit	[3]	-	[Optional, present if SDMOptions[Bit 5] = 1b] SDMReadCtrLimit value (LSB first), see Table 165
DeferOption	[1]		[Optional, present if FileOption[b3] is set] Deferral Option (see Table 165)
DeferMethod	[1]		[Optional, present if FileOption[b3] is set] Deferral Method (see Table 165)
SW1SW2	2	0x9100 0x91XX	successful execution Refer to Table 158

Table 157. Response description - GetFileSettings - Targeting FileType.Counter

Name	Length	Value	Description
FileType	1	-	File Type of the targeted file.
		0x06	Counter File
		Other values	RFU
FileOption	1	-	Options for the targeted file.
	Bit 7-2	000000b	RFU
	Bit 1-0		CommMode (see Table 14)
AccessRights	2	-	Set of access conditions for the 1st set in the file (see Section 6.10.2).
SW1SW2	2	0x9100 0x91XX	successful execution Refer to Table 158

Table 158. Error code description - GetFileSettings

Status	Value	Description
COMMAND_ABORTED	0xCA	Chained command or multiple pass command ongoing.
INTEGRITY_ERROR	0x1E	Invalid secure messaging MAC (only).
LENGTH_ERROR	0x7E	Command size not allowed.
PARAMETER_ERROR	0x9E	Parameter value not allowed.
FILE_NOT_FOUND	0xF0	File with targeted FileNo does not exist for the targeted application.

7.8.6 GetFileCounters

The detailed description of this command can be found in Section 6.10.3.2.

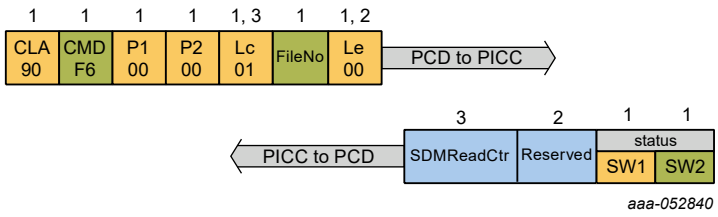


Figure 43. GetFileCounters command protocol

Table 159. Command Description - GetFileCounters

GetFileCounters	
Description:	Get file-related counters, either used for Secure Dynamic Messaging for FileType.StandardData, or from FileType.Counter.
CommMode:	CommMode.Full for SDMReadCtr retrieval on FileType.StandardData; CommMode of targeted file for FileType.Counter

Table 160. Command description - GetFileCounters

Name	Length	Value	Description
Command Header Parameters			
Cmd	1	0xF6	Command code.
FileNo	1	-	File number of the targeted file.
	Bit 7-5	000b	RFU
	Bit 4-0	Limited range	File number
Command Data Parameters			
-	-	-	No data parameters

Table 161. Response description - - Targeting FileType.StandardData with SDM enabled.

Name	Length	Value	Description
SDMReadCtr	3	Full Range	Current SDMReadCtr of the targeted file (LSB first).
Reserved	2	0x0000	RFU
SW1SW2	2	0x9100 0x91XX	successful execution Refer to Table 163

Table 162. Response description - - Targeting FileType.Counter.

Name	Length	Value	Description
FileCtr	4	Full Range	The current 32-bit value (LSB first).
SW1SW2	2	0x9100 0x91XX	successful execution Refer to Table 163

Table 163. Error code description - GetFileCounters

Status	Value	Description
COMMAND_ABORTED	0xCA	Chained command or multiple pass command ongoing
INTEGRITY_ERROR	0x1E	Invalid secure messaging MAC
LENGTH_ERROR	0x7E	Command size not allowed
PARAMETER_ERROR	0x9E	Parameter value not allowed
PERMISSION_DENIED	0x9D	PICC level (MF) is selected.
		Targeted FileType.StandardData file has no Secure Dynamic Messaging enabled.
		Targeted FileType.StandardData file has SDMCtrRet access right set to 0xF.
AUTHENTICATION_ERROR	0xAE	FileAR.SDMCtrRet not granted (while different from 0xF) for targeted File Type.StandardData file due to missing authentication or authentication with the wrong key
AUTHENTICATION_ERROR	0xAE	FileAR.Read or FileAR.ReadWrite not granted (while different from 0xF) for targeted FileType.Counter file due to missing authentication or authentication with the wrong key.
CERT_ERROR	0xCE	Active ECC-based authentication not granting FileAR.SDMCtrRet for targeted FileType.StandardData file
CERT_ERROR	0xCE	Active ECC-based authentication not granting FileAR.Read or FileAR.Read Write for targeted FileType.Counter file
FILE_NOT_FOUND	0xF0	File with targeted FileNo does not exist for the targeted application.

7.8.7 ChangeFileSettings

The detailed description of this command can be found in Section 6.10.2.3.

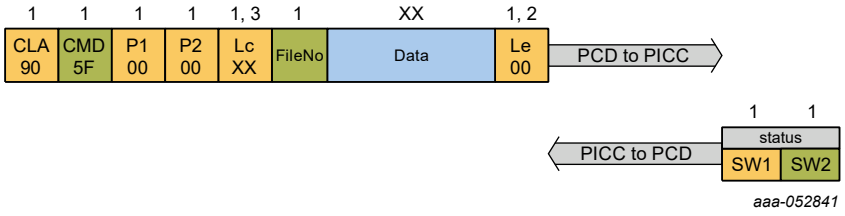


Figure 44. ChangeFileSettings command protocol

Table 164. Command summary - ChangeFileSettings

ChangeFileSettings	
Description:	Changes the access parameters and other configurations of an existing file.
CommMode:	CommMode.Full

Table 165. Command description - ChangeFileSettings

Name	Length	Value	Description
Command Header Parameters			
Cmd	1	0x5F	Command code.

Table 165. Command description - ChangeFileSettings ...continued

Name	Length	Value	Description
FileNo	1	-	File number of the targeted file.
	Bit 7-5		RFU
	Bit 4-0		File number
Command Data Parameters			
FileOption	1	-	Options for the targeted file.
	Bit 7	0b	RFU
	Bit 6		[if targeting FileNo 0x02] Secure Dynamic Messaging and Mirroring
		0b	disabled
		1b	enabled
	Bit 5-4	00b	RFU
	Bit 3	-	[if targeting FileNo 0x02] Deferred Configuration
		0b	disabled
		1b	enabled
	Bit 3	0b	[else] RFU
	Bit 2	0b	RFU
	Bit 1-0		CommMode (see Table 14).
AccessRights	2	-	Set of access conditions for the first set in the file (see Section 6.10.2).
SDMOptions	[1]	-	[Optional, present if FileOption[Bit 6] set] SDM Options
	Bit 7	-	UID (only for mirroring)
		0b	disabled
		1b	enabled
	Bit 6	-	SDMReadCtr
		0b	disabled
		1b	enabled
	Bit 5	-	SDMReadCtrLimit
		0b	disabled
		1b	enabled
	Bit 4	-	SDMENCFileData
		0b	disabled
		1b	enabled
	Bit 3	-	GPIOStatus
		0b	disabled
		1b	enabled

Table 165. Command description - ChangeFileSettings *...continued*

Name	Length	Value	Description
	Bit 2-1	00b	RFU
	Bit 0	-	Encoding mode
		1b	ASCII
SDMAccessRights	[2]	-	[Optional, present if FileOption[Bit 6] set] SDM Access Rights
	Bit 15- 12	-	SDMMetaRead access right
		0x0..0x4	Encrypted PICCData mirroring using the targeted AppKey
		0xE	Plain PICCData mirroring
		0xF	No PICCData mirroring
	Bit 11- 8	-	SDMFileRead access right
		0x0..0x4	Targeted AppKey
		0xF	No symmetric SDM for Reading
	Bit 7-4	-	SDMFileRead2 access right
		0x0..0x4	Targeted ECCPrivateKey
		0xF	No asymmetric SDM for Reading
	Bit 3-0	-	SDMCtrRet access right
		0x0..0x4	Targeted AppKey
		0xE	Free
		0xF	No Access
UIDOffset	[3]	-	[Optional, present if ((SDMOptions[Bit 7] = 1b) AND (SDMMetaRead access right = 0xE)] Mirror position (LSB first) for UID
		0x0 .. (FileSize - UIDLength)	Offset within the file
SDMReadCtrOffset	[3]	-	[Optional, present if ((SDMOptions[Bit 6] = 1b) AND (SDMMetaRead access right = 0xE)] Mirror position (LSB first) for SDMReadCtr
		0x0 .. (FileSize - SDMRead CtrLength)	Offset within the file
		0xFFFFFF	No SDMReadCtr mirroring
PICCDataOffset	[3]	-	[Optional, present if SDMMetaRead access right =0x0..0x4] Mirror position (LSB first) for encrypted PICCData
		0x0 .. (FileSize - PICCData Length)	Offset within the file

Table 165. Command description - ChangeFileSettings ...continued

Name	Length	Value	Description
GPIOStatusOffset	[3]	-	[Optional, present if (SDMOptions[Bit 3] = 1b)] Mirror position (LSB first) for GPIOStatus
		0x0 .. (FileSize-2)	Offset within the file
SDMMACInputOffset	[3]	-	[Optional, present if SDMFileRead access right != 0xF] Offset in the file where the SDM MAC computation starts (LSB first)
		0x0 .. (SDMMACOffset)	Offset within the file
SDMENCOffset	[3]	-	[Optional, present if ((SDMFileRead access right != 0xF) AND (SDMOptions[Bit 4] = 1b))] SDMENCFileData mirror position (LSB first)
		SDMMACInputOffset .. (SDMMACOffset - 32)	Offset within the file
SDMENCLength	[3]	-	[Optional, present if ((SDMFileRead access right != 0xF) AND (SDMOptions[Bit 4] = 1b))] Length of the SDMENCFileData (LSB first)
		32 .. (SDMMACOffset - SDMENCOffset)	Offset within the file, must be multiple of 32
SDMMACOffset	[3]	-	[Optional, present if SDMFileRead access right != 0xF] SDMMAC mirror position (LSB first)
		SDMMACInputOffset .. (File Size - 16)	[if (SDMFileRead access right != 0xF) AND (SDMOptions[Bit 4] = 0b)] Offset within the file
		(SDMENCOffset + SDMENCLength) .. (FileSize-16)	[if (SDMFileRead access right != 0xF) AND (SDMOptions[Bit 4] = 1b)] Offset within the file
SDMReadCtrLimit	[3]	Full range	[Optional, present if SDMOptions[Bit 5] = 1b] SDMReadCtrLimit value (LSB first)
DeferOption	[1]	-	[Optional, present if FileOption[b3] is set] Deferral Option
	Bit 7-1	'0'	RFU
	Bit 0	-	Defer SDM encryptions
		0b	disabled
		1b	enabled
DeferMethod	[1]	-	[Optional, present if FileOption[b3] is set] Deferral Method
		0x01..0x07	Number of boots (i.e. first ISO/IEC 14443-4 command)
		0xFF	ActivateConfiguration
		0x00	No deferral (expected value if DeferOption is 0x00).

Table 166. Response description - ChangeFileSettings

Name	Length	Value	Description
SW1SW2	2	0x9100 0x91XX	successful execution Refer to Table 167

Table 167. Error code description - ChangeFileSettings

Status	Value	Description
COMMAND_ABORTED	0xCA	Chained command or multiple pass command ongoing.
INTEGRITY_ERROR	0x1E	Integrity error in cryptogram. Invalid Secure Messaging MAC (only).
LENGTH_ERROR	0x7E	Command size not allowed.
PARAMETER_ERROR	0x9E	Parameter value not allowed.
		Targeted key for one of the access conditions in AccessRights or SDMAccess Rights does not exist.
		Targeted key for FileAR.SDMMetaRead or FileAR.SDMFileRead is not an existing symmetric key.
		Targeted ECCPrivateKey for FileAR.SDMFileRead2 is not existing or not enabled for ECC-based SDM.
		Trying to set FileAR.SDMMetaRead to a value different than 0xF, while both UID and SDMReadCtr mirroring are disabled.
		Trying to set FileAR.SDMMetaRead to 0xF, while enabling UID mirroring.
		Trying to set FileAR.SDMCtrRet to a value different from 0xF, while SDMReadCtr is disabled.
		SDMMAC and UID mirroring are overlapping, i.e. the following condition is not satisfied: (SDMMACOffset≥UIDOffset + UIDLength) OR (UIDOffset≥SDMMACOffset + SDMMACLength)
		SDMMAC and SDMReadCtr mirroring are overlapping, i.e. the following condition is not satisfied: (SDMMACOffset≥SDMReadCtrOffset + SDMReadCtrLength) OR (SDMReadCtrOffset≥SDMMACOffset + SDMMACLength)
		SDMMAC and PICCData mirroring are overlapping, i.e. the following condition is not satisfied: (SDMMACOffset≥PICCDataOffset + PICCDataLength) OR (PICCDataOffset≥SDMMACOffset + SDMMACLength)
		SDMMAC and GPIOStatus mirroring are overlapping, i.e. the following condition is not satisfied: (SDMMACOffset≥GPIOStatus + 3) OR (GPIOStatus≥SDMMACOffset + SDMMACLength)
		SDMSIG and GPIOStatus mirroring are overlapping, i.e. the following condition is not satisfied: (SDMMACOffset≥GPIOStatus + 3) OR (GPIOStatus≥SDMMACOffset + SDMSIGLength)
		SDMSIG and UID mirroring are overlapping, i.e. the following condition is not satisfied: (SDMMACOffset≥UIDOffset + UIDLength) OR (UIDOffset≥SDMMACOffset + SDMSIGLength)
		SDMSIG and SDMReadCtr mirroring are overlapping, i.e. the following condition is not satisfied: (SDMMACOffset≥SDMReadCtrOffset + SDMReadCtrLength) OR (SDMReadCtrOffset≥SDMMACOffset + SDMSIGLength)

Table 167. Error code description - ChangeFileSettings *...continued*

Status	Value	Description
		SDMSIG and PICCData mirroring are overlapping, i.e. the following condition is not satisfied: (SDMMACOffset≥PICCDataOffset + PICCDataLength) OR (PICCDataOffset≥SDMMACOffset + SDMSIGLength)
		SDMENCFileData and UID mirroring are overlapping, i.e. the following conditions is not satisfied: (SDMENCOffset≥UIDOffset + UIDLength) OR (UIDOffset≥SDMENCOffset + SDMENCLength)
		SDMENCFileData and SDMReadCtr mirroring are overlapping, i.e. the following condition is not satisfied: (SDMENCOffset≥SDMReadCtrOffset + SDMReadCtr Length) OR (SDMReadCtrOffset≥SDMENCOffset + SDMENCLength)
		SDMENCFileData and PICCData mirroring are overlapping, i.e. the following condition is not satisfied: (SDMENCOffset≥PICCDataOffset + PICCDataLength) OR (PICCDataOffset≥SDMENCOffset + SDMENCLength
		GPIOStatus and UID mirroring are overlapping, i.e. the following condition is not satisfied: (GPIOStatus≥UIDOffset + UIDLength) OR (UIDOffset≥GPIOStatus + 3)
		GPIOStatus and SDMReadCtr mirroring are overlapping, i.e. the following condition is not satisfied: (GPIOStatus≥SDMReadCtrOffset + SDMReadCtr Length) OR (SDMReadCtrOffset≥GPIOStatus + 3
		GPIOStatus and PICCData mirroring are overlapping, i.e. the following condition is not satisfied: (GPIOStatus≥PICCDataOffset + PICCDataLength) OR (PICCDataOffset≥GPIOStatus + 3
		GPIOStatus is overlapping with SDMENCFileData without being fully part of the plain input data area, i.e. following condition is not satisfied: (GPIOStatus + 3≤SDMENCOffset) OR (GPIOStatus≥SDMENCOffset + SDMENCLength) OR (GPIOStatus≥SDMENCOffset) AND ((GPIOStatus + 3) ≤ (SDMENCOffset + SDMENCLength/2))
		UID and SDMReadCtr mirroring are overlapping, i.e. the following condition is not satisfied: (UIDOffset≥SDMReadCtrOffset + SDMReadCtrLength) OR (SDMRead CtrOffset≥UIDOffset + UIDLength)
		Enabling Secure Dynamic Messaging encryption (SDMOptions[b4] set to 1) is not possible if FileAR.SDMFileRead = 0xF.
		Enabling Secure Dynamic Messaging encryption (SDMOptions[b4] set to 1) is not allowed if not both SDMReadCtr and UID are mirrored (i.e. SDMOptions[b7] and SDMOptions[b6] must be set to 1)
		Trying to set a SDMReadCtrLimit while not enabling SDMReadCtr.
		Trying to set a SDMReadCtrLimit, which is smaller or equal to the current SDMReadCtr.
PERMISSION_DENIED	0x9D	PICC level (MF) is selected.
		access right Change of targeted file has access conditions set to 0xF.
		Enabling Secure Dynamic Messaging (FileOption Bit 6 set to 1b) is only allowed for FileNo 0x02.
		Enabling Deferred Configuration is only allowed for FileNo 0x02.
		Trying to enable GPIOStatus while GPIO support disabled by product configuration.
FILE_NOT_FOUND	0xF0	File with targeted FileNo does not exist for the targeted application.

Table 167. Error code description - ChangeFileSettings ...continued

Status	Value	Description
AUTHENTICATION_ ERROR	0xAE	File access right Change of targeted file not granted as there is no active authentication with the required key while the access conditions is different from 0xF.
CERT_ERROR	0xCE	Active ECC-based authentication not granting FileAR.Change access rights.

7.9 Data Management

7.9.1 ReadData

The detailed description of this command can be found in Section 6.11.1.1.

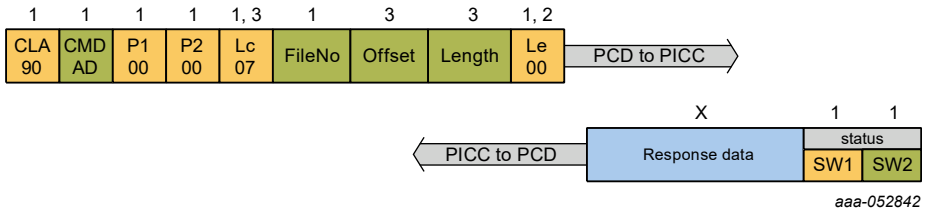


Figure 45. ReadData command protocol

Table 168. Command summary - ReadData

ReadData	
Description:	Reads data from FileType.StandardData files.
CommMode:	CommMode of targeted file.

Table 169. Command parameters description - ReadData

Name	Length	Value	Description
Command Header Parameters			
Cmd	1	0xAD	Command code.
FileNo	1	-	File number of the targeted file.
	Bit 7-5	000b	RFU
	Bit 4-0		File number
		Full Range	
Offset	3	0x000000 .. (File Size - 1)	Starting position for the read operation.
Length	3	-	Number of bytes to be read.
		0x000000	Read the entire StandardData file, starting from the position specified in the offset value.
		0x000001 .. (File Size - Offset)	
Command Data Parameters			
-	-	-	No data parameters

Table 170. Response description - ReadData

Name	Length	Value	Description
Response data	X	Full Range	Data read from the file
SW1SW2	2	0x9100 0x91XX	successful execution Refer to Table 171

Table 171. Error code description - ReadData

Status	Value	Description
COMMAND_ABORTED	0xCA	Chained command or multiple pass command ongoing
INTEGRITY_ERROR	0x1E	Invalid secure messaging MAC (only)
		SMDRdCtr overflow
LENGTH_ERROR	0x7E	Command size not allowed
PARAMETER_ERROR	0x9E	Parameter value not allowed
PERMISSION_DENIED	0x9D	Targeted file is not of FileType.StandardData.
		Read, ReadWrite and SDMFileRead (if SDM is enabled) access right of targeted StandardData file only have access conditions set to 0xF.
		Targeted file cannot be read in VCState.NotAuthenticated as the related SDMReadCtr is equal or bigger than its SDMReadCtrLimit.
		Targeted FileNo 0x01 at PICC level, while Originality Check is disabled.
		Trying to read SDMSIG while the KeyUsageCtrLimit of the targeted key entry is enabled and reached.
FILE_NOT_FOUND	0xF0	Targeted file does not exist in the targeted application
AUTHENTICATION_ERROR	0xAE	Read, ReadWrite, and SDMFileRead (if SDM enabled) access right of targeted file not granted while at least one of the access conditions is different from 0xF.
CERT_ERROR	0xCE	Active ECC-based authentication not granting the required access rights.
BOUNDARY_ERROR	0xBE	If targeting FileType.StandardData, attempt to read beyond the file boundary.

7.9.2 WriteData

The detailed description of this command can be found in Section 6.11.1.2.

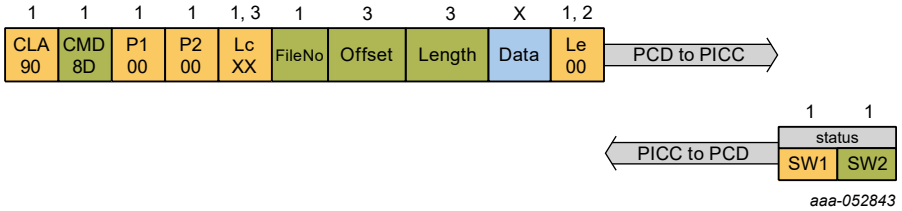


Figure 46. WriteData command protocol

Table 172. Command summary - WriteData

WriteData	
Description:	Writes data to FileType.StandardData files.
CommMode:	CommMode of targeted file.

Table 173. Command parameters description - WriteData

Name	Length	Value	Description
Command Header Parameters			
Cmd	1	0x8D	Command code.
FileNo	1	-	File number of the targeted file.
	Bit 7-5	000b	RFU
	Bit 4-0		File number
		Full range	
Offset	3	0x000000 .. (File Size - 1)	Starting position for the write operation.
Length	3	0x000001 .. (File Size - Offset)	Number of bytes to be written.
Command Data Parameters			
Data	X	Full range	Data to be written.

Table 174. Response description - WriteData

Name	Length	Value	Description
No response data parameters defined for this command			
SW1SW2	2	0x9100 0x91XX	successful execution Refer to Table 175

Table 175. Error code description - WriteData

Status	Value	Description
COMMAND_ABORTED	0xCA	Chained command or multiple pass command ongoing.

Table 175. Error code description - WriteData ...continued

Status	Value	Description
INTEGRITY_ERROR	0x1E	Invalid secure messaging MAC or encryption padding.
LENGTH_ERROR	0x7E	Command size not allowed.
PARAMETER_ERROR	0x9E	Parameter value not allowed.
PERMISSION_DENIED	0x9D	PICC level (MF) is selected.
		Targeted file is not of FileType.StandardData.
		Write and ReadWrite of targeted file only have access conditions set to 0xF.
		Targeting a StandardData file with a chained command in MAC or Full while this is not allowed.
FILE_NOT_FOUND	0xF0	Targeted file does not exist in the targeted application.
AUTHENTICATION_ ERROR	0xAE	Write and ReadWrite of targeted file not granted while at least one of the access conditions is different from 0xF.
CERT_ERROR	0xCE	Active ECC-based authentication not granting the required access rights.
BOUNDARY_ERROR	0xBE	Attempt to write beyond the file boundary as set during creation.

7.9.3 IncrementCounterFile

The detailed description of this command can be found in Section 6.11.2.1.

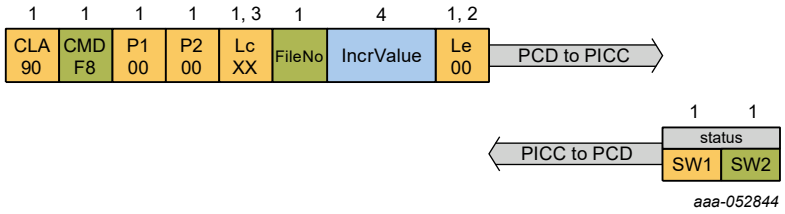


Figure 47. IncrementCounterFile command protocol

Table 176. IncrementCounterFile

IncrementCounterFile	
Description:	Increments a Counter File.
CommMode:	CommMode of targeted file.

Table 177. Command Description - IncrementCounterFile

Name	Length	Value	Description
Command Header Parameters			
CMD	1	0xF8	Command code.
FileNo	1	-	File number of the file to be incremented.
	Bit 7	'0'	Reserved
	Bit 6-5	'00'	RFU
	Bit 4-0	Full Range	File number

Table 177. Command Description - IncrementCounterFile *...continued*

Name	Length	Value	Description
Command Data Parameters			
IncrValue	4	Full Range	Value to be incremented. LSB first.

Table 178. Response description - IncrementCounterFile

Name	Length	Value	Description
No response data parameters defined for this command			
SW1SW2	2	0x9100 0x91XX	successful execution Refer to Table 179

Table 179. Error code description - IncrementCounterFile

Status	Value	Description
Resp.COMMAND_ABORTED	0xCA	Chained command or multiple pass command ongoing.
Resp.INTEGRITY_ERROR	0x1E	Invalid secure messaging MAC.
Resp.LENGTH_ERROR	0x7E	Command size not allowed.
Resp.PARAMETER_ERROR	0x9E	Parameter value not allowed.
Resp.PERMISSION_DENIED	0x9D	Not supported at PICC level.
Resp.PERMISSION_DENIED	0x9D	Targeted file is not of FileType.Counter.
Resp.PERMISSION_DENIED	0x9D	FileAR.Write and FileAR.ReadWrite of targeted file only have access conditions set to 0xF.
Resp.FILE_NOT_FOUND	0xF0	Targeted file does not exist.
Resp.AUTHENTICATION_ERROR	0xAE	FileAR.Write and FileAR.ReadWrite of targeted file not granted while at least one of the access conditions is different from 0xF.
Resp.CERT_ERROR	0xCE	Active ECC-based authentication, but CertAccessRights are not granting the required access rights.
Resp.BOUNDARY_ERROR	0xBE	File with the targeted FileNo already exists.

7.10 Crypto API

The detailed description of the usage of this command can be found in Section 6.12. The CryptoRequest restricts the maximum length of the command data to 0xFF (standard APDU), therefore, multipart command options need to be used if the input data exceeds this limit. Furthermore, the actual limit for individual command data fields may be further restricted if secure messaging must be applied.

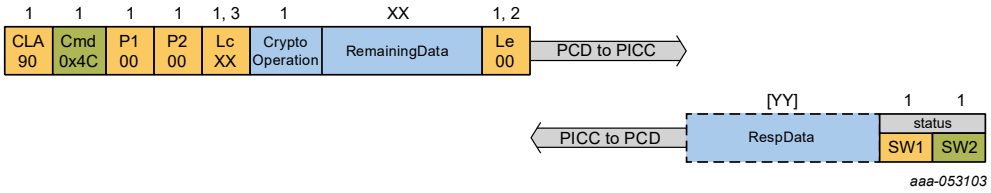


Figure 48. CryptoRequestcommand protocol

CryptoRequest	
Description:	Supports execution of various cryptographic algorithms
CommMode:	CommMode of CryptoRequest as defined by SetConfiguration 0x15.

Table 180. Command Description - CryptoRequest

Name	Length	Value	Description
Command Header Parameters			
CMD	1	0x4C	Command code.
Command Data Parameters			
Crypto Operation	1	-	The first byte of the command data specifies the operation
		0x01	SHA - for remaining command data see Section 7.10.1
		0x02	RNG - for remaining command data see Section 7.10.2
		0x03	ECC Sign - for remaining command data see Section 7.10.3
		0x04	ECC Verify - for remaining command data see Section 7.10.4
		0x05	ECC DH - for remaining command data see Section 7.10.5
		0x06	AES Enc/Dec - for remaining command data see Section 7.10.6
		0x07	Write Internal Buffer - for remaining command data see Section 7.10.9
		0x08	HMAC - for remaining command data see Section 7.10.10
		0x09	HKDF - for remaining command data see
		0x0A	AES CMAC Sign/Verfiy - for remaining command data see Section 7.10.7
		0x0B	AES AEAD Encrypt/Sign - for remaining command data see Section 7.10.8
		0x0C	AES AEAD Decrypt/Verify - for remaining command data see Section 7.10.8
		0xFD	Echo - for remaining command data see Section 7.10.12
Remaining Command Data	XX	-	

Table 181. Error Code Description - CryptoRequest

Status	Value	Description
Resp.COMMAND_ABORTED	0xCA	Chained command or multiple pass command ongoing.
Resp.INTEGRITY_ERROR	0x1E	MAC does not match data.
Resp.LENGTH_ERROR	0x7E	Command size not allowed. No MAC provided. Padding bytes wrong length
Resp.PARAMETER_ERROR	0x9E	Crypto Operation not valid
Resp.PARAMETER_ERROR	0x9E	Input source not valid
Resp.PARAMETER_ERROR	0x9E	Output destination not valid
Resp.PARAMETER_ERROR	0x9E	Parameter value not allowed
Resp.PERMISSION_DENIED	0x9D	Crypto API is disabled
Resp.PERMISSION_DENIED	0x9D	Not supported at PICC level.
Resp.PERMISSION_DENIED	0x9D	Update or finalize operation specified and no on-going multipart operation
Resp.PERMISSION_DENIED	0x9D	Access condition is 0xF
Resp.AUTHENTICATION_ERROR	0xAE	No active authentication granting the Access Condition while different from 0x0F
Resp.AUTHENTICATION_ERROR	0xAE	Slot policy does not permit operation
Resp.CERT_ERROR	0xCE	Active ECC-based authentication while Access Condition not granted while different from 0xF
Resp.BOUNDARY_ERROR	0xBE	Input source specified as an internal buffer and number of input bytes results in ‘out of bounds’ e.g. use 64 bytes from slot 5 of the TB
Resp.BOUNDARY_ERROR	0xBE	Output data does not fit output buffer

7.10.1 CryptoRequest SHA

It is possible to execute an SHA calculation using a single command or as a series of commands. Using multiple steps allows the input data to be taken from different sources.

Table 182. CryptoRequest SHA - SHA Init Operation

Name	Length	Value	Description
SHA Operation	1	0x01	Init operation
SHA Algorithm	1	0x01	SHA-256
		0x02	SHA-384
Input Data Source	1	Table 38	
Input Data Length	[1]	0xXX	Length of input data, only present when the input source is an internal buffer
Input Data	[XX]	-	Input data when the input source is the command buffer

Table 183. CryptoRequest SHA - SHA Update Operation

Name	Length	Value	Description
SHA Operation	1	0x02	Update existing SHA operation
Input Data Source	1	Table 38	
Input Data Length	[1]	0xXX	Length of input data, only present when the input source is an internal buffer
Input Data	[XX]	-	Input data when the input source is the command buffer

Table 184. CryptoRequest SHA - SHA Finalize Operation

Name	Length	Value	Description
SHA Operation	1	0x03	Finalize current SHA operation
Input Data Source	1	Table 38	
Input Data Length	[1]	0xXX	Length of input data, only present when the input source is an internal buffer
Input Data	[XX]	-	Input data when the input source is the command buffer
Result Destination	1	Table 38	

Table 185. CryptoRequest SHA - SHA One-Shot Operation

Name	Length	Value	Description
SHA Operation	1	0x04	One-shot operation
SHA Algorithm	1	0x01	SHA-256
		0x02	SHA-384
Input Data Source	1	Table 38	
Input Data Length	[1]	0xXX	Length of input data, only present when the input source is an internal buffer
Input Data	[XX]	-	Input data when the input source is the command buffer
Result Destination	1	Table 38	

Table 186. Response description - SHA Operation

Name	Length	Value	Description
Response data	[32 or 48]	-	Hash when destination is the command buffer and operation is finalize or one-shot
SW1SW2	2	0x9100 0x91XX	successful execution Refer to Table 181

7.10.2 CryptoRequest RNG

It is possible to generate random data, which is compliant with NIST SP800-90B using a 256-bit key. The Maximum number of generated bytes is 128.

Table 187. CryptoRequest RNG - RNG Operation

Name	Length	Value	Description
Num Bytes	1	0x01 - 0x80	The number of bytes to generate
Result Destination	1	Table 38	

Table 188. Response description - RNG Operation

Name	Length	Value	Description
Response data	[1 - 128]	-	Random data bytes if destination is the command buffer
SW1SW2	2	0x9100 0x91XX	successful execution Refer to Table 181 and Table 189

Table 189. Error Code Description - RNG Operation

Status	Value	Description
Resp.PERMISSION_DENIED	0x9D	Number of bytes requested is invalid

7.10.3 CryptoRequest ECC_Sign

The ECC signature generation API supports signing of a data stream or a pre-computed hash. The input may be provided in the command buffer or located in an internal buffer. The Signature shall be output to the command buffer (64 bytes of raw signature data).

Table 190. CryptoRequest ECC_Sign - ECC Sign Init Operation

Name	Length	Value	Description
ECC Sign Operation	1	0x01	Init operation
Algorithm	1	0x00	ECDSA with SHA-256
Private Key Id	1	0x00 - 0x04	Id of the ECC key pair containing the private key to use. Note a key pair must be marked as 'Crypto API Signature' ***Note:*** *A key pair must be marked as 'Crypto API Signature'*
Input Data Source	1	Table 38	
Input Data Length	[1]	0xXX	Length of input data, only present when the input source is an internal buffer
Input Data	[XX]	-	Raw data bytes, only present when input source is the command buffer

Table 191. CryptoRequest ECC_Sign - ECC Sign Update Operation

Name	Length	Value	Description
ECC Sign Operation	1	0x02	Update data to be signed

Table 191. CryptoRequest ECC_Sign - ECC Sign Update Operation...*continued*

Name	Length	Value	Description
Input Data Source	1	Table 38	
Input Data Length	[1]	0xXX	Length of input data, only present when the input source is an internal buffer
Input Data	[XX]	-	Raw data bytes, only present when input source is the command buffer

Table 192. CryptoRequest ECC_Sign - ECC Sign Finalize Operation

Name	Length	Value	Description
ECC Sign Operation	1	0x03	Finalize signature operation
Input Data Source	1	Table 38	
Input Data Length	[1]	0xXX	Length of input data, only present when the input source is an internal buffer
Input Data	[XX]	-	Raw data bytes, only present when input source is the command buffer

Table 193. CryptoRequest ECC_Sign - ECC Sign One-Shot Operation

Name	Length	Value	Description
ECC Sign Operation	1	0x04	One-shot operation
Algorithm	1	0x00	ECDSA with SHA-256
Private Key Id	1	0x00 - 0x04	Id of the ECC key pair containing the private key to use. ***Note:*** *A key pair must be marked as 'Crypto API Signature'*
Input Data Source	1	Table 38	
Input Data Length	[1]	0xXX	Length of input data, only present when the input source is an internal buffer
Input Data	[XX]	-	Raw data bytes, only present when input source is the command buffer

Table 194. CryptoRequest ECC_Sign - ECC Sign One-Shot Pre-computed Hash Operation

Name	Length	Value	Description
ECC Sign Operation	1	0x05	One-shot with pre-somputed hash operation
Algorithm	1	0x00	ECDSA with SHA-256
Private Key Id	1	0x00 - 0x04	Id of the ECC key pair containing the private key to use. Note a key pair must be marked as 'Crypto API Signature'
Input Data Source	1	Table 38	
Input Data Length	[1]	0x20	Length of input data, only present when the input source is an internal buffer
Input Data	[32]	-	Hash data bytes only present when input source is the command buffer

Table 195. Response description - ECC Sign Operation

Name	Length	Value	Description
Response data	[64]	-	Signature bytes if operation is finalize or one-shot
SW1SW2	2	0x9100 0x91XX	successful execution Refer to Table 181 and Table 196

Table 196. Error Code Description - ECC Sign Operation

Status	Value	Description
Resp.PERMISSION_DENIED	0x9D	Key id valid but key is not marked as 'Crypto API Signature'
Resp.LENGTH_ERROR	0x9D	Key usage counter limit is enabled and has been reached
Resp.LENGTH_ERROR	0x7E	Operation is one-shot with pre-computed hash and input length is not 32 bytes

7.10.4 CryptoRequest ECC_Verify

The ECC signature verification API supports verification of a data stream or data, which has already been hashed. The input may be provided in the input buffer or located in an internal buffer. The Signature to verify shall be provided in the command buffer. The signature verification successful result shall be provided as response data.

Table 197. CryptoRequest ECC_Verify - ECC Sign Init Operation

Name	Length	Value	Description
ECC Verify Operation	1	0x01	Init operation
Algorithm	1	0x00	ECDSA with SHA-256
Curve	1	0x0C	NIST 256
		0x0D	Brainpool 256
Host's Public Key	65	-	The public key to use for signature verification provided in uncompressed format
Input Data Source	1	Table 38	
Input Data Length	[1]	0xXX	Length of input data, only present when the input source is an internal buffer
Input Data	[XX]	-	Raw data bytes, only present when input source is the command buffer

Table 198. CryptoRequest ECC_Verify - ECC Verify Update Operation

Name	Length	Value	Description
ECC Verify Operation	1	0x02	Update data to be verified
Input Data Source	1	Table 38	
Input Data Length	[1]	0xXX	Length of input data, only present when the input source is an internal buffer

Table 198. CryptoRequest ECC_Verify - ECC Verify Update Operation*...continued*

Name	Length	Value	Description
Input Data	[XX]	-	Raw data bytes, only present when input source is the command buffer

Table 199. CryptoRequest ECC_Verify - ECC Verify Finalize Operation

Name	Length	Value	Description
ECC Verify Operation	1	0x03	Finalize verification operation
Signature	64	-	Signature to verify
Input Data Source	1	Table 38	
Input Data Length	[1]	0xXX	Length of input data, only present when the input source is an internal buffer
Input Data	[XX]	-	Raw data bytes, only present when input source is the command buffer

Table 200. CryptoRequest ECC_Verify - ECC Verify One-Shot Operation

Name	Length	Value	Description
ECC Verify Operation	1	0x04	One-shot verification operation
Algorithm	1	0x00	ECDSA with SHA-256
Curve	1	0x0C	NIST 256
		0x0D	Brainpool 256
Host's Public Key	65	-	The public key to use for signature verification provided in uncompressed format
Signature	64	-	Signature to verify
Input Data Source	1	Table 38	
Input Data Length	[1]	0xXX	Length of input data, only present when the input source is an internal buffer
Input Data	[XX]	-	Raw data bytes, only present when input source is the command buffer

Table 201. CryptoRequest ECC_Verify - ECC Verify One-Shot Pre-computed Hash Operation

Name	Length	Value	Description
ECC Verify Operation	1	0x05	One-shot with pre-somputed hash operation
Algorithm	1	0x00	ECDSA with SHA-256
Curve	1	0x0C	NIST 256
		0x0D	Brainpool 256
Host's Public Key	65	-	The public key to use for signature verification provided in uncompressed format i.e. leading 0x04 byte
Signature	64	-	Signature to verify

Table 201. CryptoRequest ECC_Verify - ECC Verify One-Shot Pre-computed Hash Operation...*continued*

Name	Length	Value	Description
Input Data Source	1	Table 38	
Input Data Length	[1]	0x20	Length of input data, only present when the input source is an internal buffer
Input Data	[0x20]	-	Hash data bytes only present when input source is the command buffer

Table 202. Response description - ECC Verify Operation

Name	Length	Value	Description
Response data	[2]	-	Signature verification result if operation is finalize or one-shot: 0x5A5A if successfully, otherwise 0xA5A5
SW1SW2	2	0x9100 0x91XX	successful execution Refer to Table 181 and Table 203

Table 203. Error Code Description - ECC Verify Operation

Status	Value	Description
Resp.PARAMETER_ERROR	0x9E	Public key format byte is not uncompressed 0x04

7.10.5 CryptoRequest ECC DH

The ECC Diffie-Hellman API supports the use of static keys or ephemeral keys. In addition, it allows the shared secret to be generated using a single or two-step approach. The output destination of the 32 byte shared secret shall be either the command buffer or an internal buffer.

If using a single step and the key pair Id indicates an ephemeral key then the ephemeral public key shall be output in the command buffer. The shared secret shall be output to the destination specified..

If using a two-step approach and the key pair Id indicates an ephemeral key, then the ephemeral public key shall be output in the command buffer in step 1. In the second step, the shared secret shall be output to the destination specified.

Table 204. CryptoRequest ECC_DH - ECC DH Single-step Operation

Name	Length	Value	Description
ECC DH Operation	1	0x01	One-step operation
Key Pair Id	1	0x00 - 0x04	Static key pair - the key pair must be marked as ‘Crypto API ECDH'
		0xFE	Use NIST 256 ephemeral key pair
		0xFF	Use Brainpool 256 ephemeral key pair
Shared secret destination	1	Table 38	
Public key of the Host	65	-	The host's public key to use for shared secret generation, provided in uncompressed format I.e leading 0x04 byte

Table 205. CryptoRequest ECC_DH - ECC DH Two-step Step 1

Name	Length	Value	Description
ECC DH Operation	1	0x02	Two step - first step
Key Pair Id	1	0x00 - 0x04	Static key pair - the key pair must be marked as 'Crypto API ECDH'
		0xFE	Use NIST 256 ephemeral key pair
		0xFF	Use Brainpool 256 ephemeral key pair

Table 206. CryptoRequest ECC_DH - ECC DH Two-step Step 2

Name	Length	Value	Description
ECC DH Operation	1	0x03	Two step - final step
Key Pair Id	1	0x00 - 0x04	Static key pair - the key pair must be marked as 'Crypto API ECDH'
		0xFE	Use NIST 256 ephemeral key pair
		0xFF	Use Brainpool 256 ephemeral key pair
Shared secret destination	1	Table 38	
Host's public key	65	-	The host's public key to use for shared secret generation, provided in uncompressed format I.e leading 0x04 byte

Table 207. Response description - ECC DH Operation

Name	Length	Value	Description
Card's ephemeral publickey	[65]	-	If key pair Id indicates an ephemeral key and single step or two-step step 1
Shared Secret	[32]	-	If single step or two-step step 2 and output destination is the command buffer
SW1SW2	2	0x9100 0x91XX	successful execution Refer to Table 181 and Table 208

Table 208. Error Code Description - ECC DH Operation

Status	Value	Description
Resp.PERMISSION_DENIED	0x9D	Key id valid but key is not marked as 'crypto API ECDH'
Resp.LENGTH_ERROR	0x9D	Key usage counter limit is enabled and has been reached
Resp.LENGTH_ERROR	0x9D	Two-step step 2 operation is specified and no ongoing Two-step operation
Resp.LENGTH_ERROR	0x9D	Two-step step 2 operation and key id not consistent with step 1

7.10.6 CryptoRequest AES

The AES API supports the use of static crypto API keys or keys stored in an internal buffer. The AES primitives supported by a static key are defined by the KeyPolicy set via the ChangeKey command.

Table 209. Crypto API AES Key Selection

b7	b6	b5	b4	b3	b2	b1	b0	Description
0	0	0	1	0	Key Id			Id of AES Key (must be in crypto API range: '10' – '17'), the key length from the static key
1	0	0	0	0	Slot Num			Transient buffer slot number containing the AES key, the key length shall be in the following field
1	1	0	0	Slot Num				Static buffer slot number containing the AES key, the key length shall be in the following field

The output destination for multi-part AES encryption and decryption shall always be the command buffer. For a one-shot operation, the result destination can be an internal buffer.

Table 210. Crypto API AES Key Selection - AES Enc/Dec Init Operation

Name	Length	Value	Description
AES Operation	1	0x01	Init operation
AES Primitive	1	0x03	AES-CBC Encrypt
		0x04	AES-CBC Decrypt
		0x05	AES-ECB Encrypt
		0x06	AES-ECB Decrypt
AES Key Id	1	Table 209	Id of the AES key
AES Key length	[1]	0x10 or 0x20	Length of AES key, only present when the key source is an internal buffer.
ICV Source	[1]	Table 38	Only present for CBC operations.
ICV	[16]	-	Only present for CBC operations and the ICV is in the command buffer.
Input Data Source	1	Table 38	
Input Data Length	[1]	0xXX	Length of input data, only present when the input source is an internal buffer
Input Data	[XX]	-	Raw data bytes, only present when input source is the command buffer

Table 211. Crypto API AES Key Selection - AES Enc/Dec Update Operation

Name	Length	Value	Description
AES Operation	1	0x02	Update data to be processed
Input Data Source	1	Table 38	
Input Data Length	[1]	0xXX	Length of input data, only present when the input source is an internal buffer
Input Data	[XX]	-	Raw data bytes, only present when input source is the command buffer

Table 212. Crypto API AES Key Selection - AES Enc/Dec Finalize Operation

Name	Length	Value	Description
AES Operation	1	0x03	finalize the operation
Input Data Source	1	Table 38	
Input Data Length	[1]	0xXX	Length of input data, only present when the input source is an internal buffer
Input Data	[XX]	-	Raw data bytes, only present when input source is the command buffer

Table 213. Crypto API AES Key Selection - Format of crypto API AES Enc/Dec multi-part operation response data

Name	Length	Value	Description
Output Result	[16-224]	-	Output is always present apart from first call if input data is 16 bytes or less as card stores 1 block of data until the finalize call.
SW1SW2	2	0x9100 0x91XX	successful execution Refer to Table 181 and Table 216

Table 214. Crypto API AES Key Selection - AES Enc/Dec One-Shot Operation

Name	Length	Value	Description
AES Operation	1	0x04	One-shot operation
AES Primitive	1	0x03	AES-CBC Encrypt
		0x04	AES-CBC Decrypt
		0x05	AES-ECB Encrypt
		0x06	AES-ECB Decrypt
AES Key Id	1	Table 209	Id of the AES key
AES Key length	[1]	0x10 or 0x20	Length of AES key, only present when the key source is an internal buffer.
ICV Source	[1]	Table 38	Only present for CBC operations.
ICV	[16]	-	Only present for CBC operations and the ICV is in the command buffer.
Input Data Source	1	Table 38	
Input Data Length	[1]	0xXX	Length of input data, only present when the input source is an internal buffer
Input data	[XX]	-	Raw data bytes, only present when input source is the command buffer
Result Destination	1	Table 38	

Table 215. Crypto API AES Key Selection - Format of crypto API AES Enc/Dec One-shot operation response data

Name	Length	Value	Description
Output Result	[16-224]	-	Only present when result destination is the command buffer
SW1SW2	2	0x9100 0x91XX	successful execution Refer to Table 181 and Table 216

Table 216. Error Code Description - AES Operation

Status	Value	Description
Resp.PARAMETER_ERROR	0x9E	Key id valid but key does not support AES operation key
Resp.PARAMETER_ERROR	0x9E	Total input data length is specified and the cumulative Input data bytes received in the Initialize, Update, and Finalize or One-shot operations does not match the Total Input data length field

7.10.7 CryptoRequest AES CMAC

The AES API supports the use of static crypto API keys or keys stored in an internal buffer. The AES primitives supported by a static key are defined by the KeyPolicy set via the ChangeKey command.

The CMAC Signature shall be output to the command buffer (16 bytes of raw signature data).

Table 217. CryptoRequest AES CMAC - AES CMAC Sign Init Operation

Name	Length	Value	Description
AES Operation	1	0x01	Init operation
AES Primitive	1	0x01	AES-CMAC Sign
AES Key Id	1	Table 209	Id of the AES Key
AES Key length	[1]	0x10 or 0x20	Length of AES key, only present when the key source is an internal buffer.
Input Data Source	1	Table 38	
Input Data Length	[1]	0xXX	Length of input data, only present when the input source is an internal buffer
Input Data	[XX]	-	Raw data bytes, only present when input source is the command buffer

Table 218. CryptoRequest AES CMAC - AES CMAC Sign Update Operation

Name	Length	Value	Description
AES Operation	1	0x02	Update data to be signed
Input Data Source	1	Table 38	
Input Data Length	[1]	0xXX	Length of input data, only present when the input source is an internal buffer
Input Data	[XX]	-	Raw data bytes, only present when input source is the command buffer

Table 219. CryptoRequest AES CMAC - AES CMAC Sign Finalize Operation

Name	Length	Value	Description
AES Operation	1	0x03	Finalize the signature generation operation
Input Data Source	1	Table 38	
Input Data Length	[1]	0xXX	Length of input data, only present when the input source is an internal buffer
Input Data	[XX]	-	Raw data bytes, only present when input source is the command buffer

Table 220. CryptoRequest AES CMAC - AES CMAC Sign One-shot Operation

Name	Length	Value	Description
AES Operation	1	0x04	One-shot operation
AES Primitive	1	0x01	AES-CMAC Sign
AES Key Id	1	Table 209	Id of the AES Key
AES Key length	[1]	0x10 or 0x20	Length of AES key, only present when the key source is an internal buffer.
Input Data Source	1	Table 38	
Input Data Length	[1]	0xXX	Length of input data, only present when the input source is an internal buffer
Input Data	[XX]	-	Raw data bytes, only present when input source is the command buffer

Table 221. CryptoRequest AES CMAC - Format of crypto API AES CMAC Sign response data

Name	Length	Value	Description
Output Result	[16]	-	16 bytes CMAC signature if one-shot or finalize operation
SW1SW2	2	0x9100 0x91XX	successful execution Refer to Table 181 and Table 227

The CMAC signature to verify shall be provided in the command buffer. The input data shall be provided in the command buffer or an internal buffer. The signature verification result shall be provided as response data and shall be 0x5A5A upon successful verification or 0xA5A5 if verification fails.

Table 222. CryptoRequest AES CMAC - AES CMAC Verify Init Operation

Name	Length	Value	Description
AES Operation	1	0x01	Init operation
AES Primitive	1	0x02	AES-CMAC Verify
AES Key Id	1	Table 209	Id of the AES Key
AES Key length	[1]	0x10 or 0x20	Length of AES key, only present when the key source is an internal buffer.
Input Data Source	1	Table 38	
Input Data Length	[1]	0xXX	Length of input data, only present when the input source is an internal buffer

Table 222. CryptoRequest AES CMAC - AES CMAC Verify Init Operation...*continued*

Name	Length	Value	Description
Input Data	[XX]	-	Raw data bytes, only present when input source is the command buffer

Table 223. CryptoRequest AES CMAC - AES CMAC Verify Update Operation

Name	Length	Value	Description
AES Operation	1	0x02	Update data to be verified
Input Data Source	1	Table 38	
Input Data Length	[1]	0xXX	Length of input data, only present when the input source is an internal buffer
Input Data	[XX]	-	Raw data bytes, only present when input source is the command buffer

Table 224. CryptoRequest AES CMAC - AES CMAC Verify Finalize Operation

Name	Length	Value	Description
AES Operation	1	0x03	Finalize the signature verification operation
CMAC Length	1	0x08 or 0x10	CMAC signature length
CMAC Signature	8 or 16	-	
Input Data Source	1	Table 38	
Input Data Length	[1]	0xXX	Length of input data, only present when the input source is an internal buffer
Input Data	[XX]	-	Raw data bytes, only present when input source is the command buffer

Table 225. CryptoRequest AES CMAC - AES CMAC Verify One-shot Operation

Name	Length	Value	Description
AES Operation	1	0x04	One-shot operation
AES Primitive	1	0x02	AES-CMAC Verify
AES Key Id	1	Table 209	Id of the AES Key
AES Key length	[1]	0x10 or 0x20	Length of AES key, only present when the key source is an internal buffer.
CMAC Length	1	0x08 or 0x10	CMAC signature length
CMAC Signature	8 or 16	-	
Input Data Source	1	Table 38	
Input Data Length	[1]	0xXX	Length of input data, only present when the input source is an internal buffer
Input Data	[XX]	-	Raw data bytes, only present when input source is the command buffer

Table 226. CryptoRequest AES CMAC - Format of crypto API AES CMAC Verify response data

Name	Length	Value	Description
Output Result	[2]	-	Signature verification result if one-shot or finalize operation: 0x5A5A if successful, otherwise 0xA5A5
SW1SW2	2	0x9100 0x91XX	successful execution Refer to Table 181 and Table 227

Table 227. Error Code Description - AES Operation

Status	Value	Description
Resp.PARAMETER_ERROR	0x9E	Key id valid but key does not support AES operation key
Resp.PARAMETER_ERROR	0x9E	Verify operation and MAC doesn't equal hash size
Resp.PARAMETER_ERROR	0x9E	Primitive indicates AES-CMAC verify and the CMAC length is not 8 bytes or 16 bytes
Resp.PARAMETER_ERROR	0x9E	Total input data length is specified and the cumulative Input data bytes received in the Initialize, Update, and Finalize or One-shot operations does not match the Total Input data length field.

7.10.8 CryptoRequest AES AEAD

The AES API supports the use of static crypto API keys or keys stored in an internal buffer. The AES primitives supported by a static key are defined by the KeyPolicy set via the ChangeKey command.

The output destination for multi-part AEAD shall always be the command buffer. For a one-shot operation, the result destination can be an internal static or transient buffer.

Table 228. CryptoRequest AES AEAD - AES AEAD Initialize Operation

Name	Length	Value	Description
AES Operation	1	0x01	Initialize operation
AES Primitive	1	0x07	AES-CCM Encrypt/Sign
		0x08	AES-CCM Encrypt/Sign with internally generated nonce
		0x09	AES-CCM Decrypt/Verify
		0x0A	AES-GCM Encrypt/Sign
		0x0B	AES-GCM Encrypt/Sign with internally generated nonce
		0x0C	AES-GCM Decrypt/Verify
AES Key Id	1	Table 209	Id of the AES key
AES Key length	[1]	0x10 or 0x20	Length of AES key, only present when the key source is an internal buffer.
Nonce Source	[1]	Table 38	Not present when internally generated – AES primitive 0x08 or 0x0B
Nonce length	1	0x0D	AES CCM
		0x0C - 0x3C	AES GCM
Nonce	[XX]	-	Not present when Nonce is internally generated – AES primitive 0x08 or 0x0B
AAD Source	1	Table 38	
AAD Length	1	0xXX	Number of AAD bytes.

Table 228. CryptoRequest AES AEAD - AES AEAD Initialize Operation...*continued*

Name	Length	Value	Description
AAD	[XX]	-	
Input Data Source	1	Table 38	
Input Data Length	1	0xXX	Length of input data
Input data	[XX]	-	Raw data bytes. Note all AAD data must be received before any input data can be processed.
Result Destination	[1]	Table 38	Only present when Action is 0x0C Decrypt/Verify.

Table 229. CryptoRequest AES AEAD - Format of crypto API AES AEAD Initialize operation response data

Name	Length	Value	Description
Nonce	[1]	-	Only present when Nonce is internally generated i.e. Primitive is Encrypt/Sign with internally generated Nonce 0x08 or 0x0B
Output Data	[XX]	-	Encrypted/decrypted data. The length shall be less than or equal to Input data lengths since up to 16 bytes of input data can be buffered for the next update or finalize command
SW1SW2	2	0x9100 0x91XX	successful execution Refer to Table 181 and Table 236

Table 230. CryptoRequest AES AEAD - AES AEAD Update Operation

Name	Length	Value	Description
AES Operation	1	0x02	Update AAD or Input data operation
AAD Source	1	Table 38	
AAD Length	1	0xXX	Number of AAD bytes.
AAD	[XX]	-	
Input Data Source	1	Table 38	
Input Data Length	1	0xXX	Length of input data. Note all AAD data must be received before any input data can be processed.
Input data	[XX]	-	Raw data bytes
Result Destination	1	Table 38	Only present when Action is 0x0C Decrypt/Verify.

Table 231. CryptoRequest AES AEAD - Format of crypto API AES AEAD Update operation response data

Name	Length	Value	Description
Output Data	[XX]	-	Encrypted/decrypted data. The length shall be less than or equal to Input data lengths since up to 16 bytes of input data can be buffered for the next update or finalize command
SW1SW2	2	0x9100 0x91XX	successful execution Refer to Table 181 and Table 236

Table 232. CryptoRequest AES AEAD - AES AEAD Finalize Operation

Name	Length	Value	Description
AES Operation	1	0x03	Finalize existing operation
AAD Source	1	Table 38	
AAD Length	1	0xXX	Number of AAD bytes.
AAD	[XX]	-	
Input Data Source	1	Table 38	
Input Data Length	1	0xXX	The last block of input data
Input data	[XX]	-	Note all AAD data must be received before any input data can be processed.
Tag Length	[1]	0x08 or 0x10	CCM
		0x0C or 0x10	GCM
Tag Data	[XX]	-	Only present when Action is 0x0C Decrypt/Verify.
Result Destination	1	Table 38	Only present when Action is 0x0C Decrypt/Verify.

Table 233. CryptoRequest AES AEAD - Format of crypto API AES AEAD finalize operation response data

Name	Length	Value	Description
Output Data	[XX]	-	Encrypted/decrypted data. The length shall be at minimum the input length but can be up to 16 bytes greater due to possible buffering of input data from the previous initialize or update command
Tag Data	[XX]	-	Tag data when performing an enc/sign operation i.e. Action 0x0B and AES primitive 0x08 or 0x0B
Verification Result	[2]	-	0x5A5A for successful verification, 0xA5A5 for failed verification. Only present when Action is 0x0C Decrypt/Verify.
SW1SW2	2	0x9100 0x91XX	successful execution Refer to Table 181 and Table 236

Table 234. CryptoRequest AES AEAD - AES AEAD One-Shot Operation

Name	Length	Value	Description
AES Operation	1	0x04	One-shot operation
AES Primitive	1	0x07	AES-CCM Encrypt/Sign
		0x08	AES-CCM Encrypt/Sign with internally generated nonce
		0x09	AES-CCM Decrypt/Verify
		0x0A	AES-GCM Encrypt/Sign
		0x0B	AES-GCM Encrypt/Sign with internally generated nonce
		0x0C	AES-GCM Decrypt/Verify
AES Key Id	1	Table 209	Id of the AES key
AES Key length	[1]	0x10 or 0x20	Length of AES key, only present when the key source is an internal buffer.

Table 234. CryptoRequest AES AEAD - AES AEAD One-Shot Operation...*continued*

Name	Length	Value	Description
Nonce Source	[1]	Table 38	Not present when internally generated – AES primitive 0x08 or 0x0B
Nonce length	1	0x0D	AES CCM
		0x0C - 0x3C	AES GCM
Nonce	[XX]	-	Not present when Nonce is internally generated – AES primitive 0x08 or 0x0B
AAD Source	1	Table 38	
AAD Length	1	0xXX	Number of AAD bytes.
AAD	[XX]	-	
Input Data Source	1	Table 38	
Input Data Length	1	0xXX	Length of input data
Input data	[XX]	-	Raw data bytes
Tag Length	1	0x08 or 0x10	CCM
		0x0C or 0x10	GCM
Tag Data	[XX]	-	Only present when Action is 0x0C Decrypt/Verify.
Result Destination	1	Table 38	Only present when Action is 0x0C Decrypt/Verify.

Table 235. CryptoRequest AES AEAD - Format of crypto API AES AEAD One-shot operation response data

Name	Length	Value	Description
Nonce	[1]	-	Only present when Nonce is internally generated i.e. Primitive is Encrypt/Sign with internally generated Nonce 0x08 or 0x0B
Output Data	[XX]	-	Encrypted/decrypted data. The length shall be at minimum the input length but can be up to 16 bytes greater due to possible buffering of input data from the previous initialize or update command
Tag Data	[XX]	-	Tag data when performing an enc/sign operation i.e. Action 0x0B and AES primitive 0x08 or 0x0B
Verification Result	[2]	-	0x5A5A for successful verification, 0xA5A5 for failed verification. Only present when Action is 0x0C Decrypt/Verify.
SW1SW2	2	0x9100 0x91XX	successful execution Refer to Table 181 and Table 236

Table 236. Error Code Description - AES Operation

Status	Value	Description
Resp.PARAMETER_ERROR	0x9E	Key id valid but key does not support AES operation key
Resp.PARAMETER_ERROR	0x9E	Primitive indicates CCM and the ICV isn't specified as 13 bytes
Resp.PARAMETER_ERROR	0x9E	Primitive indicates GCM and the ICV length is not in the range 12 to 60 bytes

Table 236. Error Code Description - AES Operation...*continued*

Status	Value	Description
Resp.PARAMETER_ERROR	0x9E	Total AAD length is specified and the cumulative AAD bytes received in the Initialize, Update, and Finalize or One-shot operations does not match the Total AAD length field
Resp.PARAMETER_ERROR	0x9E	Total input data length is specified and the cumulative Input data bytes received in the Initialize, Update and Finalize or One-shot operations does not match the Total Input data length field.
Resp.PARAMETER_ERROR	0x9E	Operation is AEAD CCM and the Tag field length isn't 0x08 or 0x10

7.10.9 CryptoRequest Write Internal Buffer

It is possible to write a specific value to an internal buffer using this command option. This allows data to be loaded for use within other crypto API operations.

Table 237. CryptoRequest - Write Internal Buffer Operation

Name	Length	Value	Description
Destination	1	Table 38	
Length	1		The number of bytes to write (1 byte granularity supported)
Data	XX		Data to write to the internal buffer

Table 238. Response description - Write Internal Buffer

Name	Length	Value	Description
SW1SW2	2	0x9100 0x91XX	successful execution Refer to Table 181

7.10.10 CryptoRequest HMAC

It is possible to execute an HMAC calculation using a single command or as a series of commands.
Using multiple steps allows the input data to be taken from different sources. The API uses a secure SHA implementation. A successful HMAC signature verification shall give response data 0x5A5A and a failed HMAC signature verification result shall give response data 0xA5A5.

Table 239. CryptoRequest HMAC - HMAC Operation

Name	Length	Value	Description
HMAC Operation	1	0x01	Initialize HMAC operation
		0x02	Update existing HMAC operation
		0x03	Finalize existing HMAC operation
		0x04	One-shot HMAC operation
HMAC Primitive	1	0x01	HMAC Sign
		0x02	HMAC Verify
Digest Algorithm	[1]	-	Required for Initialize and One-shot operations
		0x01	SHA256
		0x02	SHA384

Table 239. CryptoRequest HMAC - HMAC Operation...*continued*

Name	Length	Value	Description
Key Id	[1]	Table 209	Id of the HMAC key, required for Initialize and One-shot operations, otherwise absent
Key length	[1]	0x01 to 0xFF	Length of HMAC key, only present when the key source is an internal buffer.
HASH Mac	[1]	0x20 or 0x30	HASH MAC bytes. Length is equal to the Digest algorithm output length. Required for Finalizeand One-shot operations when performing HMAC Verify, otherwise absent
Input Data Source	1	Table 38	
Input Data Length	[1]	0xXX	Length of data to use (only needed if input source is an internal buffer, otherwise implied from Lc)
Input data	[XX]	-	Input data if input source is the command buffer
Result Destination	1	Table 38	Required for Finalize and One-shot sign operations, otherwise absent

Table 240. Response description - HMAC Verify Operation

Name	Length	Value	Description
Response data	2	-	Verification result: 0x5A5A if successful, otherwise 0xA5A5
SW1SW2	2	0x9100 0x91XX	successful execution Refer to Table 181 and Table 242

Table 241. Response description - HMAC Sign Operation

Name	Length	Value	Description
Response data	[32 or 48]	-	Hmac signature if output destination is the command buffer
SW1SW2	2	0x9100 0x91XX	successful execution Refer to Table 181 and Table 242

Table 242. Error Code Description - HMAC Operation

Status	Value	Description
Resp.PARAMETER_ERROR	0x9E	Key id valid but key not an HMAC key
Resp.PARAMETER_ERROR	0x9E	Verify operation and MAC doesn't equal hash size

7.10.11 CryptoRequest HKDF

HKDF, as defined in RFC5869, requires execution of the extract operation followed by the expand operation. The API uses a secure SHA implementation.

Table 243. CryptoRequest HKDF - HKDF Extract and Expand Operation

Name	Length	Value	Description
HKDF Operation	1	0x00	Extract and expand
Digest Algorithm	1	0x01	SHA256
		0x02	SHA384
Key Id	1	Table 209	Initial Key Material (IKM)
Key length	[1]	0x01 to 0xFF	Length of HMAC key, only present when the key source is an internal buffer.
Salt Source	1	Table 38	
Salt Length	1	0x00 to 0x80	Length of salt – If salt length is 0 then a zero salt value of hash length bytes shall be used
Salt Data	[XX]	-	Salt data if salt source is the command buffer
Info Source	1	Table 38	
Info Length	1	0x01 to 0x50	Length of context and info data. Note that zero length is not supported.
Info Data	[XX]	-	Context data if context source is the command buffer
Result Destination	1	Table 38	
Result Length	1	0x01 to 0xEF	Number of bytes to output

Table 244. CryptoRequest HKDF - HKDF Expand Operation

Name	Length	Value	Description
HKDF Operation	1	0x01	Expand
Digest Algorithm	1	0x01	SHA256
		0x02	SHA384
Key Id	1	Table 209	Pseudorandom key (PRK)
Key length	[1]	0x20 or 0x30	Length of PRK, only present when the key source is an internal buffer. Length must be equal to the Hash byte length.
Info Source	1	Table 38	
Info Length	1	0x01 to 0x50	Length of context and info data. ***Note:*** *Zero length is not supported.*
Info Data	[XX]	-	Context data if context source is the command buffer
Result Destination	1	Table 38	
Result Length	1	0x01 to 0xEF	Number of bytes to output

Table 245. Response description - HKDF Operation

Name	Length	Value	Description
Response data	[1 - 239]	-	HKDF result if output destination is the command buffer
SW1SW2	2	0x9100 0x91XX	successful execution Refer to Table 181 and Table 246

Table 246. Error Code Description - HKDF Operation

Status	Value	Description
Resp.PARAMETER_ERROR	0x9E	Key id valid but key not an HKDF key
Resp.LENGTH_ERROR	0x7E	Length of command data not consistent with the length fields specified

Application Remark:

HKDF expand operation fails with error message 910E if Info Length is zero.

7.10.12 CryptoRequest Echo

It is possible to have the device echo the command data provided to it. This may be useful to verify system setup.

Table 247. CryptoRequest Echo - Echo Operation

Name	Length	Value	Description
Additional Data Bytes	0x00 - 0xFE	-	Additional bytes to echo

Table 248. Response description - Echo Operation

Name	Length	Value	Description
Echo Operation Byte	1	0xFD	
Additional bytes	[1 - 254]	-	Addition bytes received in the command data
SW1SW2	2	0x9100 0x91XX	successful execution Refer to Table 181

7.11 GPIO Management

7.11.1 ManageGPIO

The detailed description of this command can be found in Section 6.13.1.

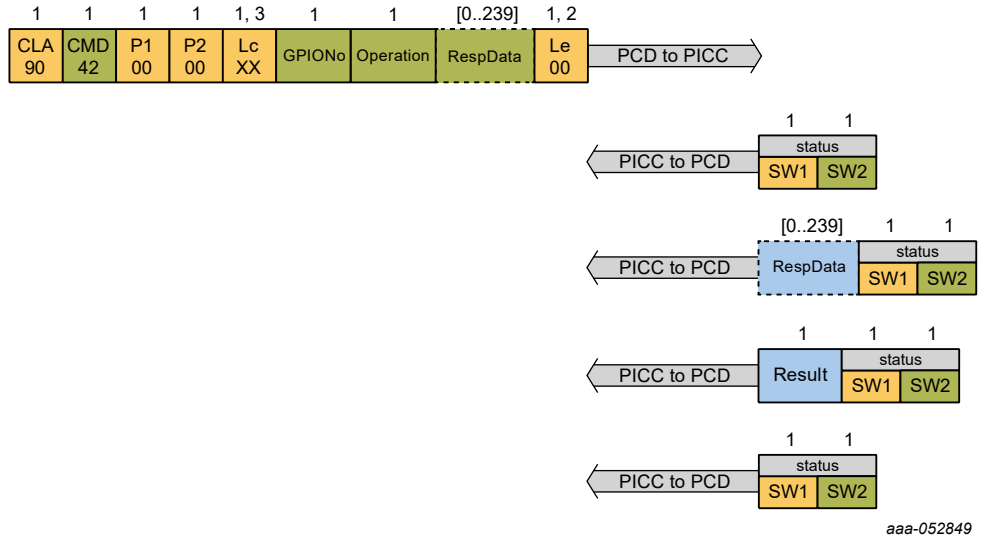


Figure 49. ManageGPIO command protocol

Table 249. ManageGPIO

ManageGPIO	
Description:	Manages the GPIO output.
CommMode:	CommMode of ManageGPIO as defined by SetConfiguration 0x11.

Table 250. Command Description - ManageGPIO

Name	Length	Value	Description
Command Header Parameters			
CMD	1	0x42	Command code.
GPIONo	1	-	GPIO Number
		0x00	GPIO1
		0x01	GPIO2
Operation	1	-	Targeted operation
	Bit 7	-	RFU
	Bit 6-2	'00000'	[if GPIOxMode is output] RFU
	Bit 1-0	-	[if GPIOxMode is output] GPIO Control
		'00'	CLEAR: clear the GPIO state to LOW (not driven).
		'01'	SET: set the GPIO State to HIGH (driven).

Table 250. Command Description - ManageGPIO *...continued*

Name	Length	Value	Description
		'10'	TOGGLE: toggle the GPIO State.
		'11'	RFU
RespData	[0 .. 239]	Full range	[Optional, present if GPIOXMode is output AND Operation[b7] == '1' AND issued over I2C]

Table 251. Response description - ManageGPIO [else]

Name	Length	Value	Description
No response data parameters defined for this command			
SW1SW2	2	0x9100 0x91XX	successful execution Refer to Table 252

Table 252. Error code description - ManageGPIO

Status	Value	Description
Resp.COMMAND_ABORTED	0xCA	Chained command or multiple pass command ongoing.
Resp.INTEGRITY_ERROR	0x1E	Integrity error in cryptogram or invalid secure messaging MAC
Resp.LENGTH_ERROR	0x7E	Command size not allowed.
Resp.PARAMETER_ERROR	0x9E	Parameter value not allowed.
Resp.PERMISSION_DENIED	0x9D	Not supported at PICC level.
Resp.PERMISSION_DENIED	0x9D	Targeting GPIO1 while it is not configured for output or downstream power out.
Resp.PERMISSION_DENIED	0x9D	Triggering execution of MEASURE while down-stream power out was already enabled.
Resp.AUTHENTICATION_ERROR	0xAE	No active authentication granting the ManageGPIOAccess
Resp.CERT_ERROR	0xCE	Active ECC-based authentication while ManageGPIOAccess

7.11.2 ReadGPIO

The detailed description of this command can be found in Section 6.13.2.

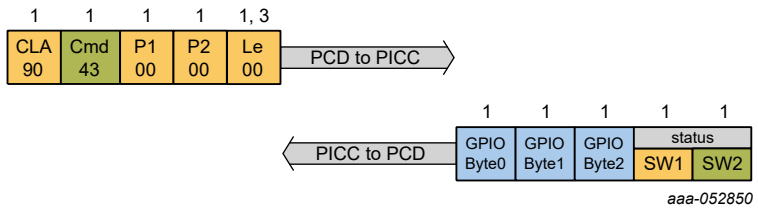


Figure 50. ReadGPIO command protocol

Table 253. ReadGPIO

ReadGPIO	
Description:	Returns the GPIO statuses.

Table 253. ReadGPIO...continued

ReadGPIO	
CommMode:	CommMode of ReadGPIO as defined by SetConfiguration 0x11.

Table 254. Command Description - ReadGPIO

Name	Length	Value	Description
Command Header Parameters			
Cmd	1	0x43	Command code.

Table 255. Response description - ReadGPIO

Name	Length	Value	Description
GPIOByte0	1	-	GPIOStatus bytes as defined in Table 41.
		0x43	[if TagTamper] Close
		0x4F	[if TagTamper] Open
		0x49	[else] Invalid
GPIOByte1	1	-	GPIOStatus bytes as defined in Table 41.
		0x43	[if TagTamper] Close
		0x4F	[if TagTamper] Open
		0x48	[if Input or Output] High
		0x4C	[if Input or Output] Low
		0x49	[else] Invalid
GPIOByte2	1	-	GPIOStatus bytes as defined in Table 41.
		0x48	[if Input or Output] High
		0x4C	[if Input or Output] Low
		0x49	[else] Invalid
SW1SW2	2	0x9100 0x91XX	successful execution Refer to Table 256

Table 256. Error code description - ReadGPIO

Status	Value	Description
COMMAND_ABORTED	0xCA	Chained command or multiple pass command ongoing.
INTEGRITY_ERROR	0x1E	Integrity error in cryptogram or invalid secure messaging MAC
LENGTH_ERROR	0x7E	Command size not allowed.
PARAMETER_ERROR	0x9E	Parameter value not allowed.
PERMISSION_DENIED	0x9D	Not supported at PICC level.
PERMISSION_DENIED	0x9D	ReadGPIOAccessCondition is configured for no access (0x0F).
AUTHENTICATION_ERROR	0xAE	No active authentication granting the ReadGPIOAccessCondition while different from 0x0F
CERT_ERROR	0xCE	Active ECC-based authentication while ReadGPIOAccessCondition not granted while different from 0xF.

7.12 ISO7816-4 Support

7.12.1 ISOSelectFile

The detailed description of this command can be found in Section 6.15.1.4.

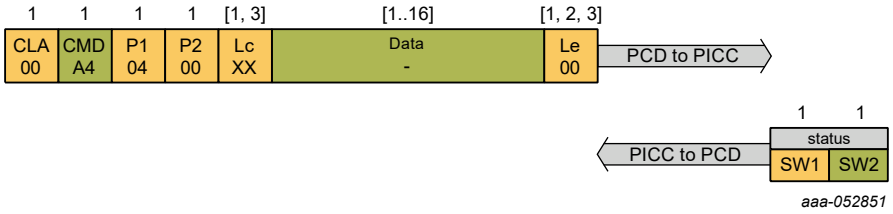


Figure 51. ISOSelectFile command protocol

Table 257. Command summary - ISOSelectFile

ISOSelectFile	
Description:	Select an application or file
CommMode:	N/A

Table 258. Command description - ISOSelectFile

Name	Length	Value	Description
CLA	1	0x00	
INS	1	0xA4	
P1	1	-	Selection Control
		0x00	Select MF, DF or EF, by file identifier
		0x01	Select child DF
		0x02	Select EF under the current DF, by file identifier
		0x03	Select parent DF of the current DF
		0x04	Select by DF name, see [3]

Table 258. Command description - ISOSelectFile *...continued*

Name	Length	Value	Description
P2	1	-	Option
		0x00	Return FCI template: data stored in the file with ID 0x1F should be returned
		0x0C	No response data: no FCI should be returned
Lc	[1, 3]	0x00 .. 0x10	Length of subsequent data field
Data	[1..16]	-	Reference
		Empty	[if P1 == 0x00 OR P1 == 0x03] Select MF
		Full range	[if P1 == 0x00 OR P1 == 0x01 OR P1== 0x02] Select with the given file identifier
		Full range	[if P1 == 0x04] Select DF with the given DF name
Le	[1, 2, 3]	Full range	Empty or length of expected response

Table 259. Response description - ISOSelectFile

Name	Length	Value	Description
Data	[X]	Full range	[Optional] FCI stored in file ID 31 of the DF
SW1SW2	2	0x9000 0xXXXX	successful execution Refer to Table 260

Table 260. Error code description - ISOSelectFile

SW1|SW2	Value	Description
ISO6700	0x6700	Wrong or inconsistent APDU length.
ISO6985	0x6985	Wrapped chained command or multiple pass command ongoing.
ISO6A82	0x6A82	Application or file not found, currently selected application remains selected.
ISO6A86	0x6A86	Wrong parameter P1 and/or P2
ISO6A87	0x6A87	Wrong parameter Lc inconsistent with P1-P2
ISO6E00	0x6E00	Wrong CLA

7.12.2 ISOReadBinary

The detailed description of this command can be found in Section 6.15.1.5.

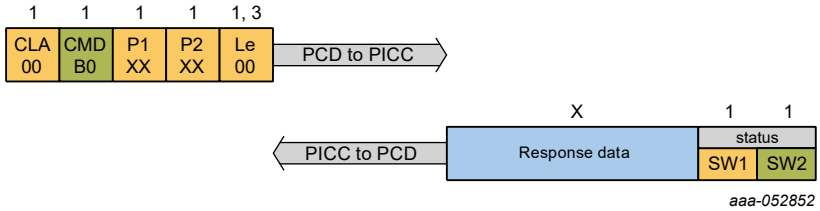


Figure 52. ISOReadBinary command protocol

Table 261. Command summary - ISOReadBinary

ISOReadBinary	
Description:	Read from a data file
CommMode:	N/A

Table 262. Command description - ISOReadBinary

Name	Length	Value	Description
CLA	1	0x00	
INS	1	0xB0	
P1	1		ShortFile ID/Offset
	Bit 7		Encoding
		1b	P1[Bit 6..5] are RFU. P1[Bit 4..0] encode a short ISO FileID. P2[Bit 7..0] encode an offset from zero to 255.
		0b	P1 - P2 (15 bits) encode an offset from zero to 32767.
	Bit 6-5	00b	[if P1[7] == 1b] RFU
	Bit 4-0		[if P1[7] == 1b] short ISO FileID
		0x00	Targeting currently selected file.
		0x01 .. 0x1E	Targeting and selecting file referenced by the given short ISO FileID.
		0x1F	RFU
	Bit 6-0	(see P2)	[if P1[7] == 0b] Most significant bits of Offset
P2	1	0x000000 .. (File Size - 1)	Offset (see above)
Le	1, 3	-	The number of bytes to be read from the file.
		0x000000	Read the entire data file, starting from the position specified in the offset value.
		0x000001 .. 0xFFFFFF	If bigger than (FileSize - Offset), the entire StandardData file starting from the offset position is returned.
		Full range	

Table 263. Response description - ISOReadBinary

Name	Length	Value	Description
Data	X	-	Data read.
SW1SW2	2	0x9000 0xXXXX	successful execution Refer to Table 264

Table 264. Error code description - ISOReadBinary

SW1|SW2	Value	Description
ISO6700	0x6700	Wrong or inconsistent APDU length.
ISO6982	0x6982	Security status not satisfied: no access allowed as Read and ReadWrite access rights are different from 0xE and SDMFileRead (if SDM enabled) access right is set to 0xF.
		Security status not satisfied: SDMReadCtr overflow.
		Security status not satisfied: Targeted file cannot be read in VCState.Not Authenticated as the related SDMReadCtr is equal or bigger than its SDMRead CtrLimit.
		Security status not satisfied: AuthenticatedAES not allowed.
		Security status not satisfied: AuthenticatedECC not allowed.
ISO6985	0x6985	Wrapped chained command or multiple pass command ongoing. No file selected. Attempt to read outside file boundaries. Targeted file with ISO FileID 0xEF01 at PICC level, while Originality Check is disabled. Trying to readSDMSIG while the KeyUsageCtrLimit of the targeted key entry is enabled and reached.
ISO6A82	0x6A82	File not found
ISO6A86	0x6A86	Wrong parameter P1 and/or P2
ISO6E00	0x6E00	Wrong CLA

7.12.3 ISOUpdateBinary

The detailed description of this command can be found in Section 6.15.1.6.

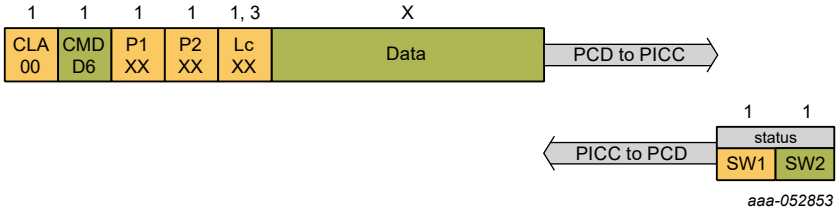


Figure 53. ISOUpdateBinary command protocol

Table 265. Command summary - ISOUpdateBinary

ISOUpdateBinary	
Description:	Write to a data file
CommMode:	N/A

Table 266. Command description - ISOUpdateBinary

Name	Length	Value	Description
CLA	1	0x00	
INS	1	0xD6	
P1	1		ShortFile ID/Offset
	Bit 7		RFU
		1b	P1[Bit 6..5] are RFU. P1[Bit 4..0] encode a short ISO FileID. P2[Bit 7..0] encode an offset from zero to 255.
		0b	P1 - P2 (15 bits) encode an offset from zero to 32767.
	Bit 6-5	00b	[if P1[7] == 1b] RFU
	Bit 4-0		[if P1[7] == 1b] short ISO FileID
		0x00	Targeting currently selected file.
		0x01 .. 0x1E	Targeting and selecting file referenced by the given short ISO FileID.
		0x1F	RFU
	Bit 6-0	(see P2)	[if P1[7] == 0b] Most significant bits of Offset
P2	1	0x000000 .. (File Size - 1)	Offset (see above)
Lc	1, 3	0x000001 .. (File Size - Offset)	Length of subsequent data field
Data	X	Full range	Data to be written

Table 267. Response description - ISOUpdateBinary

Name	Length	Value	Description
No response data parameters defined for this command			
SW1SW2	2	0x9000 0xXXXX	successful execution Refer to Table 268

Table 268. Error code description - ISOUpdateBinary

SW1|SW2	Value	Description
ISO6700	0x6700	Wrong or inconsistent APDU length.
ISO6982	0x6982	Security status not satisfied: only free write with Write or ReadWrite equal to 0xE is allowed. Security status not satisfied: AuthenticatedAES not allowed. Security status not satisfied: AuthenticatedECC not allowed.
ISO6985	0x6985	Wrapped chained command or multiple pass command ongoing. No file selected. Attempt to write beyond the file boundary as set during creation.
ISO6A82	0x6A82	File not found
ISO6A86	0x6A86	Wrong parameter P1 and/or P2
ISO6E00	0x6E00	Wrong CLA

8 Limiting values

Table 269. Limiting values

In accordance with the Absolute Maximum Rating System (IEC 60134). Voltages are referenced to VSS (ground = 0 V).

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
V_{CC}	supply voltage		-0.3	-	2	V
V_I	input voltage	Any supply pad	-0.3	-	2	V
I_I	input current	pads SDA, SCL	-	-	10	mA
I_O	output current	pads SDA, SCL	-	-	10	mA
I_{LU}	latch-up current	$V_I < 0$ V or $V_I > V_{CC}$	-	-	100	mA
V_{ESD}	electrostatic discharge voltage	human body model (HBM)[1] pads V_{CC}, V_{SS}, SDA, SCL, GPIO1, GPIO2	-	-	+/- 2	kV
V_{ESD}	electrostatic discharge voltage	human body model (HBM)[1]	-	-	+/- 4	kV
V_{ESD}	electrostatic discharge voltage	charged device model (CDM)[2] pads V_{CC}, V_{SS}, SDA, SCL, GPIO1, GPIO2	-	-	+/- 500	V
P_{tot}	total power dissipation	[3]	-	-	40	mW
T_{stg}	storage temperature		-65	-	150	°C

[1] According to ANSI/ESDA/JEDEC JS-001
[2] According to ANSI/ESDA/JEDEC JS-002
[3] Depending on the appropriate thermal resistance of the package.

CAUTION

This device is sensitive to ElectroStatic Discharge (ESD). Observe precautions for handling electrostatic sensitive devices.

Such precautions are described in the *ANSI/ESD S20.20, IEC/ST 61340-5, JESD625-A* or equivalent standards.

9 Recommended operating conditions

A30 is characterized by its specified operating supply voltage range of 1 V to 2 V.

Table 270. Recommended operating conditions

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
V_{CC}	supply voltage	norminal Supply voltage	1	-	2	V
V_I	DC input voltage on digital inputs and digital I/O pads	[1]	1 V + 10 %	-	V_{CC} + 0.3 V	V
H	field strength	contactless interface operation	1.5	-	7.5	A/m
T_{amb}	operating ambient temperature	[2]	-40	-	105	°C

[1] The supply voltage operating range of 1 V to 2 V requires internal supply elevation for the supply voltage range of 1 V to 1.62 V.
The supply voltage mode is automatically selected during boot-up based on internal supply voltage measurement.
To avoid continues activation and deactivation of the internal supply voltage elevation the external supply voltage of 1.55 V to 1.62 V should be avoided as performance degradation or resets might occur in this supply voltage range due to internal supply voltage switching. Performance degradation or chip resets might lead to timeouts during I^2C communication. Therefore it is recommended that the host would continue to retry the read for a preset number of times in case of timeouts and after that it will go to recovery mode trying with interface/chip reset and even if there is no response, returns with an error for the application to reopen the session.
The V_{CC} supply voltage rise time impacts the power consumption. V_{CC} supply voltage ramp times <600 µs to 1.8 V lead to higher power consumption as the device boots in voltage elevation mode. For V_{CC} supply voltages >1.62 V the supply voltage ramp shall therefore >600 µs.
The reference design recommendations of 100 nF cpacitor close to VCC/VSS pin must be followed. The minimum V_{CC} rise time (0 % - 100 %) is larger than 25 µs.

[2] All product properties and values specified within this data sheet are only valid within the operating ambient temperature range.

10 Characteristics

10.1 DC characteristics

Measurement conventions

Testing measurements are performed at the contact pads of the device under test. All voltages are defined with respect to the ground contact pad VSS. All currents flowing into the device are considered positive.

10.1.1 General-purpose I/O interface

Table 271. Electrical DC characteristics of GPIO1/2

V_{CC} = 1 V to 2 V (V_{SS} = 0 V; T_{amb} = -40 °C to 105 °C, unless otherwise specified
External pullup resistor 20 kΩ to V_{CC} assumed. The worst case test condition for parameter V_{OH} is present at minimum V_{CC}.

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
V_{IH}	HIGH level input voltage		0.7 x V_{CC}	-	V_{CC} + 0.3	V
V_{IL}	LOW level input voltage		-0.3	-	0.25 x V_{CC}	V
I_{IH}	HIGH level input current in "weak pullup" input mode	0.7 V_{CC} ≤ V_I ≤ V_{CC} Test conditions for the maximum absolute value: I_{IH}(max): V_I = 0.7 V_{CC}, V_{CC} = $V_{CC(max)}$	-	-1	-20	µA
I_{IL}	LOW level input current	0 V ≤ V_I ≤ 0.3 V_{CC}; Test conditions for the maximum absolute value: $I_{IL(max)}$: V_I = 0 V, V_{CC} = $V_{CC(max)}$	-	-1	-50	µA
I_I	Input current in "weak pullup" input mode	0 V ≤ V_I ≤ V_{CC}; Test conditions for the maximum absolute value: II(max):V_I = 0 V, V_{CC} = $V_{CC(max)}$	0	-	-50	µA
I_{ILIH}	Leakage input current at input voltage beyond V_{CC} in "weak pullup" input mode	V_{CC} < V_I ≤ V_{CC} + 0.3 V; -40 °C ≤ T_{amb} ≤ 105 °C; Test conditions: V_I = V_{CC} + 0.3 V; V_{CC} = $V_{CC(max)}$; T_{amb} = 105 °C	-	-	20	µA
I_{ILIL}	Leakage input current at input voltage below V_{SS} in "weak pullup" input mode	-0.3 V ≤ V_I < 0 V; -40 °C ≤ T_{amb} ≤ 30 °C Test conditions: V_I = -0.3 V; V_{CC}= $V_{CC(max)}$; T_{amb} = 30 °C	-	-	-50	µA
V_{OH}	HIGH level output voltage	I_{OH} = -20 µA	0.7 x V_{CC}	-	-	V
V_{OL}	LOW level output voltage	I_{OL} = 1 mA I_{OL} = 0.5 mA	-	-	0.3 0.7 x V_{CC}	V

Conditions:

V_{CC} = 1 V to 2 V (V_{SS} = 0 V; T_{amb} = -40 °C to 105 °C, unless otherwise specified

External pullup resistor 20 kΩ to V_{CC} assumed. The worst case test condition for parameter V_{OH} is present at minimum V_{CC}.

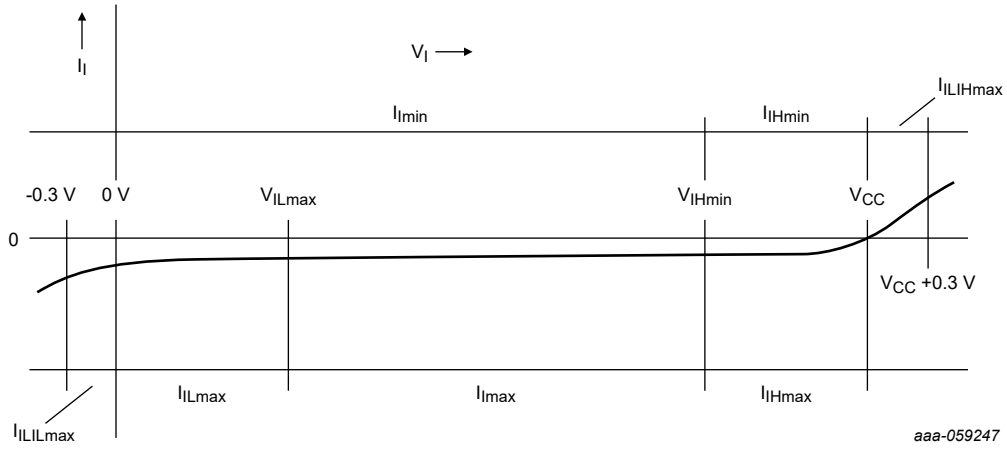


Figure 54. Input characteristics of GPIO1/2

10.1.2 I²C interface

Table 272. Electrical DC characteristics of I²C

V_{CC} = 1 V to 2 V (V_{SS} = 0 V; T_{amb} = -40 °C to 105 °C, unless otherwise specified
Pads SCL, SDA are in open-drain mode

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
V_{IH}	HIGH level input voltage		0.7 x V_{CC}	-	V_{CC} + 0.3	V
V_{IL}	LOW level input voltage		-0.3	-	0.25 x V_{CC}	V
V_{HYS}	input hysteresis voltage		0.081	-	-	V
$V_{OL(OD)}$	Low-level output voltage(open-drain mode)	I_{OL} = 3 mA	0	-	0.4	V
$I_{OL(OD)}$	Low-level output current(open-drain mode)	V_{CC} ≥1.1 V	0.6	-	-	mA
I_{WPU}	weak pullup current	V_{CC} ≥ 1.1 V	-	-180	-	µA
I_{ILIH}	leakage input current high level	V_{SDA} = 3.6 V, V_{SCL} = 3.6 V	-	0.27	15	µA

10.1.3 Power Consumption

Table 273. Electrical characteristics of IC supply voltage V_{CC}

V_{SS} = 0 V; T_{amb} = -40 °C to 105 °C, unless otherwise specified

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
V_{CC}	supply voltage range		1	-	2	V
I_{DD}	supply current high-performance mode, CPU halted and AES or ECC cryptographic in operation		-	-	15	mA
	supply current low-power processing mode, CPU in Idle mode and AES or ECC cryptographic in operation		-	-	0.65	mA
	supply current Halt mode		-	-	5	µA
	supply current Off state		-	-	0.25	µA

10.2 AC characteristics

Table 274. Authentication application timing

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
t_{DIT}	Initialization time from V_{CC} applied or wake from HALT mode		-	-	1	ms
t_{AUTH1}	Authentication time, with contact, SIGMA-I protocol		-	-	500	ms
t_{AUTH2}	Authentication time, with contactless, with ID1 antenna		-	-	90	ms

Table 275. Nonvolitile memory timing characteristics

V_{CC} = 1 V to 2 V; V_{SS} = 0 V; T_{amb} = -40 °C to 105 °C, unless otherwise specified

Symbol	Parameter	Conditions	Min	Typ[1]	Max	Unit
t_{EEP}	FLASH erase + program time[2]		-	-	2.3	ms
t_{EEE}	FLASH program time		-	-	0.9	ms
t_{EEW}	FLASH erase time		-	-	1.4	ms
t_{EER}	FLASH data retention time	T_{amb} = 55 °C	25	-	-	years
N_{EEC}	FLASH endurance (maximum number of programming cycles applied to the whole memory block performed by NXP static and dynamic wear leveling algorithm)		20×10^6	100×10^6	-	cycles

[1] Typical values are only referenced for information. They are subject to change without notice.
[2] The given value specifies physical access times of FLASH memory only.

Table 276. Electrical AC characteristics of SDA, SCL

V_{CC} = 1 V to 2 V; V_{SS} = 0 V; T_{amb} = -40 °C to 105 °C, unless otherwise specified [1]
SCL, SDA pads in open-drain mode.

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
tr_{IO}[2][3]	I/O Input rise time	Input/reception mode	-	-	1	µs
tf_{IO}[2][4]	I/O Input fall time	Input/reception mode	-	-	1	µs
tf_{OIO}	I/O Output fall time	Output/transmission mode; C_L= 30 pF	-	-	0.3	µs
f_{CLK}	External clock frequency in I^2C applications	t_{CLKW}, T_{amb} and V_{CC} within specified limits	-	-	1	MHz
C_{PIN}	Pin capacitances SDA, SCL	Test f = 1 MHz; T_{amb} = 25 °C	-	-	10.5	pF
P_{OUT}	maximum output power in power harvesting mode at GPIO1		-	-	10	mW

[1] All appropriately marked values are typical values and only referenced for information. They are subject to change without notice.
[2] maximum recommended load 5pF
[3] t_r is defined as rise time between 30 % and 70 % of the signal amplitude.
[4] t_f is defined as fall time between 70 % and 30 % of the signal amplitude.

10.3 I^2C Bus Timings

The A30 I^2C bus timing parameters are in accordance to the NXP I^2C bus specification, see Section 13.

10.4 EMC/EMI

EMC and EMI resistance according to IEC 61967-4, see Section 13.

11 Package information

A30 is either offered as Wafer Level Chip-Scale Package (WLCSP), or HVQFN.

11.1 WLCSP 16

A30 is provided in a four by four ball grid Wafer Level Chip-Scale Package (WLCSP):

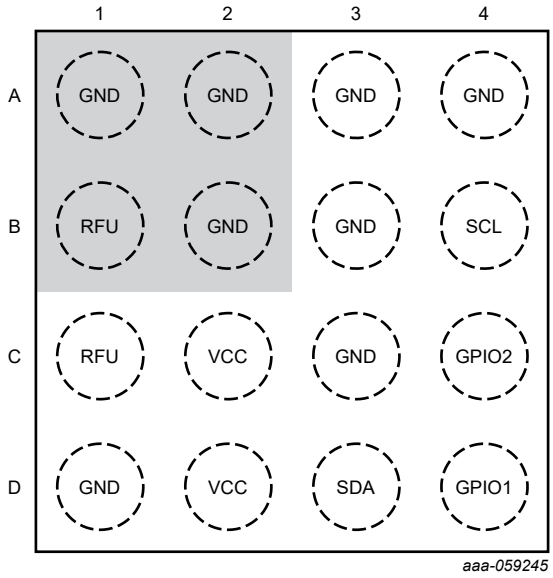


Figure 55. Package outline WLCSP (Top view)

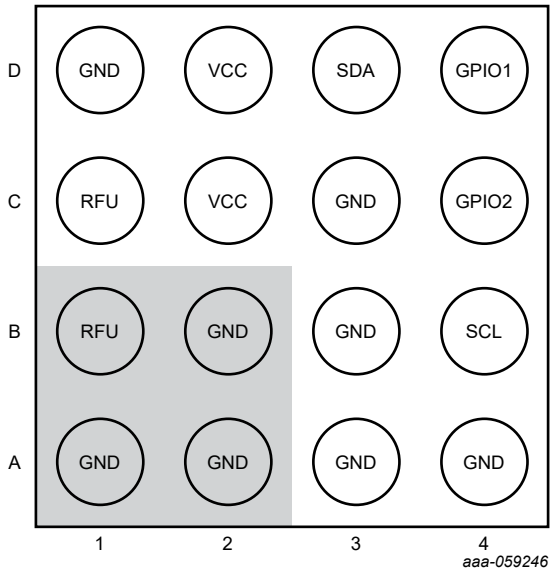


Figure 56. Package outline WLCSP (Bottom view)

WLCSP thickness is <= 0.5 mm with a ball pitch is 0.35 mm. A detailed description including pins can be found in "Delivery Specification [11]"

11.2 HVQFN 20

A30 is provided in HVQFN:

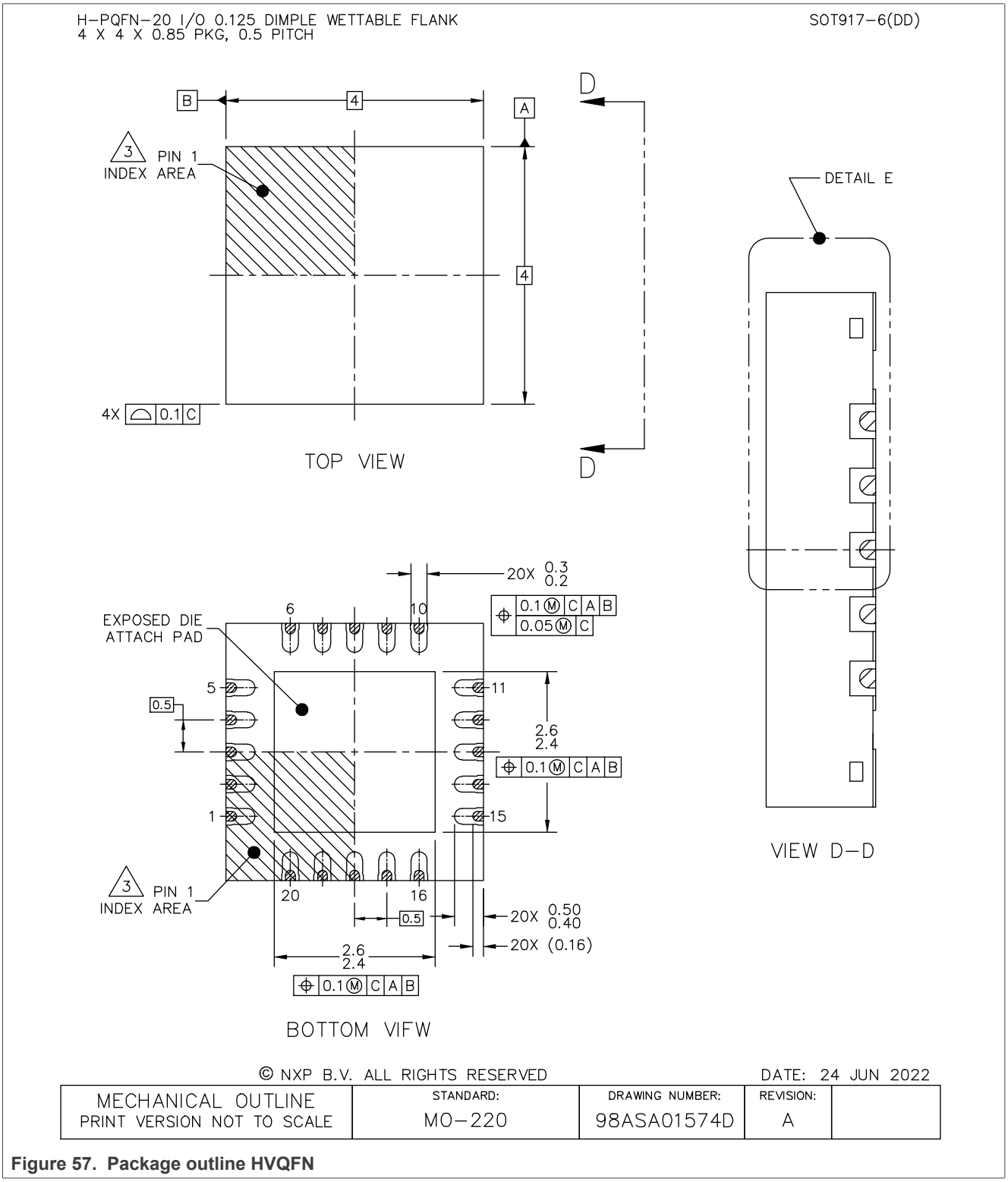


Figure 57. Package outline HVQFN

HVQFN thickness is 0.85 mm with a pitch is 0.5 mm. A detailed description can be found in "Delivery Specification [11]"

12 Abbreviations

Table 277. Abbreviations

Acronym	Description
AES	Advanced Encryption Standard
APDU	Application Protocol Data unit
AppKey	Application Key
AppMasterKey	Application Master Key
API	Application Programming Interface
ASCII	American Standard Code for Information Interchange
ATQA	Answer to Request A
ATS	Answer to Select
CA	Certificate Authority
C-APDU	Command APDU
CBC	Cipher Block Chaining
CC	Capability Container
CCM	Counter with Cipher Block Chaining Message Authentication Code (CBC-MAC)
CID	Channel Identifier
CLA	Class
CMAC	Cipher-based Message Authentication Code
CmdCtr	Command Counter
CRC	Cyclic Redundancy Check
DF	Dedicated File (Application)
EAL	Evaluation Assurance Level
ECB	Electronic Code Book mode
ECC	Error Correcting Code
ECDH	Elliptic-curve Diffie Hellman
EF	Elementary File (File)
FCI	File Control Information
FSC	Frame Size for proximity Card (according to ISO/IEC 14443-4)
GPIO	General-Purpose Input/Output
HWDT	Halt WatchDog Timer
INS	INStruction byte (according to ISO/IEC 7816-4)
IV	Initialization Vector
KDF	Key Derivation Function
LSB	Least Significant Byte
MAC	Message Authentication Code
MCU	Microcontroller Unit

Table 277. Abbreviations...*continued*

Acronym	Description
MF	Master File
MSB	Most Significant Byte
NDEF	NFC Data Exchange Format
NFC	Near-Field Communication
NVM	Non-Volatile Memory
OID	Object IDentifier
PCB	Printed-Circuit Board
PCD	Proximity Coupling Device (Contactless Reader)
PCDCap	Proximity Coupling Device Capabilities
PD	Proximity Device, used as synonym for the PICC
PDCap	Proximity Device Capabilities
PICC	Proximity IC Card
PICCData	PICC data targeted for mirroring (e.g. UID, SDMReadCtr)
PKI	Public Key Infrastructure
POR	power-on-reset
PPS	Protocol Parameter Select
PRF	Pseudo-Random Function
PST	Power-Saving Time-out
RATS	Request for Answer To Select
RC	Return Code
RFU	Reserved for Future Use
RNG	Random Number Generator
SAK	Select Acknowledge
SDA	Serial Data
SDM	Secure Dynamic Messaging
SDMCtrRet	SDM Counter Retrieval, access right for GetFileCounters
SDMENCFileData	Refers to the encrypted part of data in the NDEF file
SDMFileRead	SDM File Reading, key/access setting for Secure Dynamic Messaging
SDMFileReadKey	Refers to the AppKey which is used for SDM MAC calculation
SDMMAC	Refers to the MAC calculated over response
SDMMetaRead	SDM Meta Reading, specifies PICCData encryption key or plain mirroring
SDMMetaReadKey	Refers to the AppKey which is used for SDM encryption of PICCData
SDMReadCtr	SDM Read Counter, counting number of interactions with a PICC
SesAuthENCKey	Session key for encryption
SesAuthMACKey	Session key for MACing
SP	Special Publication

Table 277. Abbreviations...*continued*

Acronym	Description
SPI	Serial Peripheral Interface
SUN	Secure Unique NFC
SV	Session Vector, input for session key calculation
SW	Status Word
TI	Transaction Identifier
TT	Tag Tamper
TTCurrStatus	Current status of the Tag Tamper loop
TTPermStatus	Permanently stores an Open status on the Tag Tamper loop
UID	Unique IDentifier
URI	Uniform Resource Identifier
WLCSP	Wafer Level Chip Sale Package

13 References

[1] **User Guidance Manual**
A30 Information on Guidance and Operation, Doc. No. UM9763**[1]

[2] **ISO/IEC 14443-3:2018**
Identification cards -- Contactless integrated circuit cards -- Proximity cards -- Part 3: Initialization and anti-collision

[3] **ISO/IEC 14443-4:2018**
Identification cards -- Contactless integrated circuit cards -- Proximity cards -- Part 4: Transmission protocol

[4] **ISO/IEC 7816-4:2020**
Identification cards – Integrated circuit cards – Part 4: Organization, security and commands for interchange

[5] **FIPS PUB 197**
FEDERAL INFORMATION PROCESSING STANDARDS PUBLICATION, ADVANCED ENCRYPTION STANDARD (AES), National Institute of Standards and Technology, 2001 November 26

[6] **NIST Special Publication 800-38A**
National Institute of Standards and Technology (NIST). Recommendation for BlockCipher Modes of Operation.
http://csrc.nist.gov/publications/nistpubs/800-38a/sp800-38a.pdf

[7] **NIST Special Publication 800-38B**
National Institute of Standards and Technology (NIST). Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication.
http://csrc.nist.gov/publications/nistpubs/800-38B/SP_800-38B.pdf

[8] **ISO/IEC 9797-1:1999**
Information technology – Security techniques – Message Authentication Codes (MACs) – Part 1: Mechanisms using a block cipher.

[9] **NIST Special Publication 800-108**
National Institute of Standards and Technology (NIST). Recommendation for key derivation using pseudorandom functions.

[10] **IEEE Std 802.3-2008**
IEEE Standard for Information technology - Telecommunications and information exchange between systems - Local and metropolitan area networks - Specific requirements Part 3: Carrier sense multiple access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications.

[11] **Data sheet addendum**
A30 - Delivery specification, Document number AD9772**[1]

[12] **Certicom Research. Sec 1**
Elliptic curve cryptography. Version 2.0, May 2009.

[13] **Product data sheet**
NXP Semiconductors, NTAG213/215/216: NFC Forum Type 2 Tag compliant IC with 144/504/888 bytes user memory, Document number 2653**[1]

[14] **NFC Forum: Type 4 Tag - Technical Specification**
NFC Forum: Type 4 Tag - Technical Specification - Version 1.0 - [T4T] - 2016.07.26, 07 2016.

[15] **User manual**
UM10204 I2C-bus specification and user manual, Rev. 7, 10 2021.

[16] **GlobalPlatform**
Globalplatform technology - apdu transport over spi / i2c - version 1.0. Version 1.0, January 2020

[17] **National Institute of Standards and Technology (NIST)**

[1] ** … document version number

Federal Information Processing Standard (FIPS) 180- 4: Secure Hash Standard (SHS). NIST FIPS PUB 180-4, August 2015.

[18] **ISO/IEC 8825-1:2015**
ISO JTC 1/SC 6. Information technology - ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER). ISO/IEC 8825-1:2015, November 2015.

[19] **ISO/IEC 9798-3:2019**
ISO JTC 1/SC 27. Information technology – Security techniques – Entity authentication – Part 3: Mechanisms using digital signature techniques. ISO/IEC 9798-3:2019, 2019.

[20] **IEEE Std 802.3-2008**
IEEE Computer Society. IEEE Standard for Information Technology - Telecommunications and information exchange between systems - Local and metropolitan area networks - Specific requirements Part 3: Carrier sense multiple access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications. IEEE Std 802.3-2008, December 2008.

[21] **Proposal for: Functionality classes for random number generators**
A proposal for: Functionality classes for random number generators, Wolfgang Killmann, T-Systems GEI GmbH, Werner Schindler, Bundesamt für Sicherheit in der Informationstechnik (BSI), Version 2.0, 18 September 2011

[22] **BSI-CC-PP-0084-2014**
Security IC Platform Protection Profile with Augmentation Packages, Registered and Certified by Bundesamt für Sicherheit in der Informationstechnik (BSI) under the reference BSI-CC-PP-0084-2014, Version 1.0, 13 January 2014.

[23] **FIPS PUB 186-5**
FIPS PUB 186-5 (Draft): Digital Signature Standard (DSS), Federal Information Processing Standards Publication, US Department of Commerce/National Institute of Standards and Technology, October 2019.

[24] **FIPS PUB 198-1**
FIPS PUB 198-1: The Keyed-Hash Message Authentication Code (HMAC), Federal Information Processing Standards Publication 198-1, US Department of Commerce/ National Institute of Standards and Technology, July 2008

[25] **NIST SP 800-38C**
NIST SP 800-38C: Recommendation for Block Cipher Modes of Operation: The CCM Mode for Authentication and Confidentiality, Morris Dworkin, National Institute of Standards and Technology, May 2004.

[26] **NIST SP 800-38D**
NIST SP 800-38D: Recommendation for Block Cipher Modes of Operation: Galois/ Counter Mode (GCM) and GMAC, Morris Dworkin, National Institute of Standards and Technology, November 2007.

[27] **NIST SP 800-56A**
NIST SP 800-56A Revision 3: Recommendation for Pair-Wise Key-Establishment Schemes Using Discrete Logarithm Cryptography, National Institute of Standards and Technology, April 2018.

[28] **RFC 5869**
RFC 5869: HMAC-based Extract-and-Expand Key Derivation Function (HKDF), Internet Engineering Task Force (IETF), Request For Comments, May 2010.

[29] **ISO/IEC 9594-8**
ISO/IEC 9594-8:2020 Information technology - Open systems interconnection - Part 8: The Directory: Public key and attribute certificate frameworks - Ninth edition, 11 2020.

14 Note about the source code in the document

Example code shown in this document has the following copyright and BSD-3-Clause license:

Copyright 2023-2025 NXP Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials must be provided with the distribution.
3. Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

15 Revision history

Table 278. Revision history

Document ID	Release date	Description
A30 v.3.0 [1]	27 January 2025	Initial version for the public release

[1] Previous versions are not published

Legal information

Data sheet status

Document status[1][2]	Product status[3]	Definition
Objective [short] data sheet	Development	This document contains data from the objective specification for product development.
Preliminary [short] data sheet	Qualification	This document contains data from the preliminary specification.
Product [short] data sheet	Production	This document contains the product specification.

[1] Please consult the most recently issued document before initiating or completing a design.
[2] The term 'short data sheet' is explained in section "Definitions".
[3] The product status of device(s) described in this document may have changed since this document was published and may differ in case of multiple devices. The latest product status information is available on the Internet at URL https://www.nxp.com.

Definitions

Draft — A draft status on a document indicates that the content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included in a draft version of a document and shall have no liability for the consequences of use of such information.

Short data sheet — A short data sheet is an extract from a full data sheet with the same product type number(s) and title. A short data sheet is intended for quick reference only and should not be relied upon to contain detailed and full information. For detailed and full information see the relevant full data sheet, which is available on request via the local NXP Semiconductors sales office. In case of any inconsistency or conflict with the short data sheet, the full data sheet shall prevail.

Product specification — The information and data provided in a Product data sheet shall define the specification of the product as agreed between NXP Semiconductors and its customer, unless NXP Semiconductors and customer have explicitly agreed otherwise in writing. In no event however, shall an agreement be valid in which the NXP Semiconductors product is deemed to offer functions and qualities beyond those described in the Product data sheet.

Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms and conditions of commercial sale of NXP Semiconductors.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors and its suppliers accept no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Limiting values — Stress above one or more limiting values (as defined in the Absolute Maximum Ratings System of IEC 60134) will cause permanent damage to the device. Limiting values are stress ratings only and (proper) operation of the device at these or any other conditions above those given in the Recommended operating conditions section (if present) or the Characteristics sections of this document is not warranted. Constant or repeated exposure to limiting values will permanently and irreversibly affect the quality and reliability of the device.

Terms and conditions of commercial sale — NXP Semiconductors products are sold subject to the general terms and conditions of commercial sale, as published at https://www.nxp.com/profile/terms, unless otherwise agreed in a valid written individual agreement. In case an individual agreement is concluded only the terms and conditions of the respective agreement shall apply. NXP Semiconductors hereby expressly objects to applying the customer's general terms and conditions with regard to the purchase of NXP Semiconductors products by customer.

No offer to sell or license — Nothing in this document may be interpreted or construed as an offer to sell products that is open for acceptance or the grant, conveyance or implication of any license under any copyrights, patents or other industrial or intellectual property rights.

Quick reference data — The Quick reference data is an extract of the product data given in the Limiting values and Characteristics sections of this document, and as such is not complete, exhaustive or legally binding.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

Suitability for use in non-automotive qualified products — Unless this document expressly states that this specific NXP Semiconductors product is automotive qualified, the product is not suitable for automotive use. It is neither qualified nor tested in accordance with automotive testing or application requirements. NXP Semiconductors accepts no liability for inclusion and/or use of non-automotive qualified products in automotive equipment or applications.

In the event that customer uses the product for design-in and use in automotive applications to automotive specifications and standards, customer (a) shall use the product without NXP Semiconductors' warranty of the product for such automotive applications, use and specifications, and (b) whenever customer uses the product for automotive applications beyond NXP Semiconductors' specifications such use shall be solely at customer's own risk, and (c) customer fully indemnifies NXP Semiconductors for any liability, damages or failed product claims resulting from customer design and use of the product for automotive applications beyond NXP Semiconductors' standard warranty and NXP Semiconductors' product specifications.

HTML publications — An HTML version, if available, of this document is provided as a courtesy. Definitive information is contained in the applicable document in PDF format. If there is a discrepancy between the HTML document and the PDF document, the PDF document has priority.

Translations — A non-English (translated) version of a document, including the legal information in that document, is for reference only. The English version shall prevail in case of any discrepancy between the translated and English versions.

Security — Customer understands that all NXP products may be subject to unidentified vulnerabilities or may support established security standards or specifications with known limitations. Customer is responsible for the design and operation of its applications and products throughout their lifecycles to reduce the effect of these vulnerabilities on customer's applications and products. Customer's responsibility also extends to other open and/or proprietary technologies supported by NXP products for use in customer's applications. NXP accepts no liability for any vulnerability. Customer should regularly check security updates from NXP and follow up appropriately.

Customer shall select products with security features that best meet rules, regulations, and standards of the intended application and make the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP.

NXP has a Product Security Incident Response Team (PSIRT) (reachable at PSIRT@nxp.com) that manages the investigation, reporting, and solution release to security vulnerabilities of NXP products.

NXP B.V. — NXP B.V. is not an operating company and it does not distribute or sell products.

Trademarks

Notice: All referenced brands, product names, service names, and trademarks are the property of their respective owners.

NXP — wordmark and logo are trademarks of NXP B.V.

Bluetooth — the Bluetooth wordmark and logos are registered trademarks owned by Bluetooth SIG, Inc. and any use of such marks by NXP Semiconductors is under license.

DESFire — is a trademark of NXP B.V.

I2C-bus — logo is a trademark of NXP B.V.

Matter, Zigbee — are developed by the Connectivity Standards Alliance. The Alliance's Brands and all goodwill associated therewith, are the exclusive property of the Alliance.

MIFARE — is a trademark of NXP B.V.

NTAG — is a trademark of NXP B.V.

SmartMX — is a trademark of NXP B.V.

Tables

Tab. 1. Ordering information 4
Tab. 2. A30 pin configuration 6
Tab. 3. I2C communication interface parameters 7
Tab. 4. ISO/IEC 7816-4 command fields 9
Tab. 5. ISO/IEC 7816-4 response fields 10
Tab. 6. SIGMA-I Session Keys 13
Tab. 7. SIGMA-I Message Types 13
Tab. 8. Asymmetric authentication Protocols Payload Encodings 14
Tab. 9. A30 as SIGMA-I responder 15
Tab. 10. A30 as SIGMA-I initiator 16
Tab. 11. SIGMA-I Session Key Sizes 17
Tab. 12. ECC-basedcard-unilateral authentication 21
Tab. 13. When to use which authentication command 22
Tab. 14. Supported communication modes 28
Tab. 15. PICCData: plain encoding and lengths 33
Tab. 16. PICCDataTag 34
Tab. 17. Access condition values coded on 4 bits 40
Tab. 18. ACMap encoding 41
Tab. 19. Application access rights, specified via DFName 42
Tab. 20. Application access rights, specified via DFName 43
Tab. 21. Manufacturer characteristics used as card version 43
Tab. 22. SetConfiguration options list 45
Tab. 23. ProtocolOptions 51
Tab. 24. GPIOxConfig 52
Tab. 25. GPIOxPadCtrl 53
Tab. 26. Supported memory configurations 55
Tab. 27. Supported key types 56
Tab. 28. Keys at application level 56
Tab. 29. GetKeySettings Key Groups 59
Tab. 30. Certificate Cache Example 63
Tab. 31. X.509 Certificate Wrap Encoding 67
Tab. 32. Set of Access condition coded on 2 bytes 69
Tab. 33. FileAR.SDMMetaRead values 70
Tab. 34. FileAR.SDMFileRead values 70
Tab. 35. FileAR.SDMFileRead2 values 70
Tab. 36. FileAR.SDMFileRead and FileAR.SDMFileRead2 combinations 71
Tab. 37. Command list associated with access rights 71
Tab. 38. Crypto API Data Source/Destination Selection 80
Tab. 39. Crypto API Slot Usage Policy Options 80
Tab. 40. Crypto API Policy Supported Algorithms 80
Tab. 41. ReadGPIO response 82
Tab. 42. APDUs 91
Tab. 43. Command summary - ISOGeneralAuthenticate 93
Tab. 44. Command description - ISOGeneralAuthenticate 93
Tab. 45. Response description - ISOGeneralAuthenticate 93
Tab. 46. Error code description - ISOGeneralAuthenticate 93
Tab. 47. Command summary - ISOInternalAuthenticate 94
Tab. 48. Command Description - ISOInternalAuthenticate 94
Tab. 49. Response description - ISOInternalAuthenticate 95
Tab. 50. Error code description - ISOInternalAuthenticate 95
Tab. 51. Command summary - AuthenticateEV2First ... 96
Tab. 52. Command description - AuthenticateEV2First - Part1 97
Tab. 53. Response description - AuthenticateEV2First - Part1 97
Tab. 54. Error code description - AuthenticateEV2First - Part1 97
Tab. 55. Command description - AuthenticateEV2First - Part2 98
Tab. 56. Response description - AuthenticateEV2First - Part2 98
Tab. 57. Error code description - AuthenticateEV2First - Part2 98
Tab. 58. Command summary - AuthenticateEV2NonFirst 99
Tab. 59. Command description - AuthenticateEV2NonFirst - Part1 99
Tab. 60. Response description - AuthenticateEV2NonFirst - Part1 99
Tab. 61. Error code description - AuthenticateEV2NonFirst - Part1 100
Tab. 62. Command description - AuthenticateEV2NonFirst - Part2 100
Tab. 63. Response description - AuthenticateEV2NonFirst - Part2 100
Tab. 64. Error code description - AuthenticateEV2NonFirst - Part2 100
Tab. 65. Command summary - ProcessSM 101
Tab. 66. Command Description - ProcessSM 101
Tab. 67. Response Description - ProcessSM 101
Tab. 68. Error code description - ProcessSM 101
Tab. 69. Command summary - ProcessSM_Apply 102
Tab. 70. Command Description - ProcessSM_Apply ...102
Tab. 71. Response Description - ProcessSM_Apply ... 103
Tab. 72. Error code description - ProcessSM_Apply ... 103
Tab. 73. Command summary - ProcessSM_Remove . 103
Tab. 74. Command Description - ProcessSM_ Remove 104
Tab. 75. Response Description - ProcessSM_ Remove 104
Tab. 76. Error code description - ProcessSM_ Remove 104
Tab. 77. Command summary - FreeMem 105
Tab. 78. Command description - FreeMem 105
Tab. 79. Response description - FreeMem - OPERATION_OK 105

Tab. 80. Error code description - FreeMem 105
Tab. 81. Command Description - SetConfiguration 106
Tab. 82. Command description - SetConfiguration 106
Tab. 83. Response description - SetConfiguration 107
Tab. 84. Error code description - SetConfiguration 107
Tab. 85. Command summary - GetConfiguration 108
Tab. 86. Command Description - GetConfiguration 109
Tab. 87. Response description - GetConfiguration 109
Tab. 88. Error code description - GetConfiguration 109
Tab. 89. Command summary - ActivateConfiguration 110
Tab. 90. Command Description - ActivateConfiguration 110
Tab. 91. Response description - ActivateConfiguration 110
Tab. 92. Error code description - ActivateConfiguration 111
Tab. 93. Command summary - GetVersion 111
Tab. 94. Command parameters description - GetVersion - Part1 111
Tab. 95. Response description - GetVersion - Part1 ... 112
Tab. 96. Command parameters description - GetVersion - Part2 112
Tab. 97. Response description - GetVersion - Part2 ... 112
Tab. 98. Command parameters description - GetVersion - Part3 113
Tab. 99. Response description - GetVersion - Part3 ... 113
Tab. 100. Error code description - GetVersion 114
Tab. 101. Command summary - GetCardUID 114
Tab. 102. Command parameters description - GetCardUID 114
Tab. 103. Response description - GetCardUID 114
Tab. 104. Error code description - GetCardUID 115
Tab. 105. Command summary - ChangeKey 115
Tab. 106. Command description - ChangeKey 115
Tab. 107. Response description - ChangeKey 117
Tab. 108. Error code description - ChangeKey 117
Tab. 109. Command Description - GetKeySettings 118
Tab. 110. Command description - GetKeySettings 118
Tab. 111. Response description - GetKeySettings - [No Option byte provided] 119
Tab. 112. Response description - GetKeySettings - [Option = 0x00] CryptoRequestKey's meta-data 119
Tab. 113. Response description - GetKeySettings - [Option = 0x01] ECCPrivateKey's meta-data 119
Tab. 114. Response description - GetKeySettings - [Option = 0x02] CARootKey's meta-data 120
Tab. 115. Error code description - GetKeySettings 120
Tab. 116. Command Description - GetKeyVersion 121
Tab. 117. Command parameters description - GetKeyVersion 121
Tab. 118. Response description - GetKeyVersion 121
Tab. 119. Error code description - GetKeyVersion 121
Tab. 120. ManageKeyPair 122
Tab. 121. Command Description - ManageKeyPair 122
Tab. 122. Response description - ManageKeyPair 124
Tab. 123. Error code description - ManageKeyPair 124
Tab. 124. ManageCARootKey 125
Tab. 125. Command Description - ManageCARootKey 125
Tab. 126. Response description - ManageCARootKey 126
Tab. 127. Error code description - ManageKeyPair 126
Tab. 128. Command Description - ManageCertRepo 127
Tab. 129. ManageCertRepo - Create Certificate Repository 128
Tab. 130. ManageCertRepo - Load Certificate 128
Tab. 131. ManageCertRepo - Load Certificate Mapping info 129
Tab. 132. ManageCertRepo - Reset Certificate Repository 129
Tab. 133. ManageCertRepo - Error Conditions 129
Tab. 134. Command Description - ReadCertRepo 130
Tab. 135. ReadCertRepo - Response Data Format for Metadata 130
Tab. 136. ReadCertRepo - Response Data Format for Certificate 131
Tab. 137. Error Code Description - ReadCertRepo 131
Tab. 138. Command Description - CreateStdDataFile .. 132
Tab. 139. Command description - CreateStdDataFile ... 132
Tab. 140. Response description - CreateStdDataFile ... 133
Tab. 141. Error code description - CreateStdDataFile ... 133
Tab. 142. CreateCounterFile 134
Tab. 143. Command Description - CreateCounterFile ... 134
Tab. 144. Response description - CreateCounterFile 135
Tab. 145. Error code description - CreateCounterFile ... 135
Tab. 146. Command Description - GetFileIDs 135
Tab. 147. Command description - GetFileIDs 135
Tab. 148. Response description - GetFileIDs 136
Tab. 149. Error code description - GetFileIDs 136
Tab. 150. Command Description - GetISOFileIDs 137
Tab. 151. Command description - GetISOFileIDs 137
Tab. 152. Response description - GetISOFileIDs 137
Tab. 153. Error code description - GetISOFileIDs 138
Tab. 154. Command Description - GetFileSettings 138
Tab. 155. Command description - GetFileSettings 138
Tab. 156. Response description - GetFileSettings - Targeting FileType.StandardData 139
Tab. 157. Response description - GetFileSettings - Targeting FileType.Counter 140
Tab. 158. Error code description - GetFileSettings 140
Tab. 159. Command Description - GetFileCounters 141
Tab. 160. Command description - GetFileCounters 141
Tab. 161. Response description - - Targeting FileType.StandardData with SDM enabled. ... 141
Tab. 162. Response description - - Targeting FileType.Counter. 141
Tab. 163. Error code description - GetFileCounters 142
Tab. 164. Command summary - ChangeFileSettings 142
Tab. 165. Command description - ChangeFileSettings 142
Tab. 166. Response description - ChangeFileSettings ..146
Tab. 167. Error code description - ChangeFileSettings 146
Tab. 168. Command summary - ReadData 148

Tab. 169. Command parameters description - ReadData 148
Tab. 170. Response description - ReadData 149
Tab. 171. Error code description - ReadData 149
Tab. 172. Command summary - WriteData 150
Tab. 173. Command parameters description - WriteData 150
Tab. 174. Response description - WriteData 150
Tab. 175. Error code description - WriteData 150
Tab. 176. IncrementCounterFile 151
Tab. 177. Command Description - IncrementCounterFile 151
Tab. 178. Response description - IncrementCounterFile 152
Tab. 179. Error code description - IncrementCounterFile 152
Tab. 180. Command Description - CryptoRequest 153
Tab. 181. Error Code Description - CryptoRequest 154
Tab. 182. CryptoRequest SHA - SHA Init Operation 154
Tab. 183. CryptoRequest SHA - SHA Update Operation 155
Tab. 184. CryptoRequest SHA - SHA Finalize Operation 155
Tab. 185. CryptoRequest SHA - SHA One-Shot Operation 155
Tab. 186. Response description - SHA Operation 155
Tab. 187. CryptoRequest RNG - RNG Operation 156
Tab. 188. Response description - RNG Operation 156
Tab. 189. Error Code Description - RNG Operation 156
Tab. 190. CryptoRequest ECC_Sign - ECC Sign Init Operation 156
Tab. 191. CryptoRequest ECC_Sign - ECC Sign Update Operation 156
Tab. 192. CryptoRequest ECC_Sign - ECC Sign Finalize Operation 157
Tab. 193. CryptoRequest ECC_Sign - ECC Sign One-Shot Operation 157
Tab. 194. CryptoRequest ECC_Sign - ECC Sign One-Shot Pre-computed Hash Operation 157
Tab. 195. Response description - ECC Sign Operation 158
Tab. 196. Error Code Description - ECC Sign Operation 158
Tab. 197. CryptoRequest ECC_Verify - ECC Sign Init Operation 158
Tab. 198. CryptoRequest ECC_Verify - ECC Verify Update Operation 158
Tab. 199. CryptoRequest ECC_Verify - ECC Verify Finalize Operation 159
Tab. 200. CryptoRequest ECC_Verify - ECC Verify One-Shot Operation 159
Tab. 201. CryptoRequest ECC_Verify - ECC Verify One-Shot Pre-computed Hash Operation 159
Tab. 202. Response description - ECC Verify Operation 160
Tab. 203. Error Code Description - ECC Verify Operation 160
Tab. 204. CryptoRequest ECC_DH - ECC DH Single-step Operation 160
Tab. 205. CryptoRequest ECC_DH - ECC DH Two-step Step 1 161
Tab. 206. CryptoRequest ECC_DH - ECC DH Two-step Step 2 161
Tab. 207. Response description - ECC DH Operation .. 161
Tab. 208. Error Code Description - ECC DH Operation 161
Tab. 209. Crypto API AES Key Selection 162
Tab. 210. Crypto API AES Key Selection - AES Enc/Dec Init Operation 162
Tab. 211. Crypto API AES Key Selection - AES Enc/Dec Update Operation 162
Tab. 212. Crypto API AES Key Selection - AES Enc/Dec Finalize Operation 163
Tab. 213. Crypto API AES Key Selection - Format of crypto API AES Enc/Dec multi-part operation response data 163
Tab. 214. Crypto API AES Key Selection - AES Enc/Dec One-Shot Operation 163
Tab. 215. Crypto API AES Key Selection - Format of crypto API AES Enc/Dec One-shot operation response data 164
Tab. 216. Error Code Description - AES Operation 164
Tab. 217. CryptoRequest AES CMAC - AES CMAC Sign Init Operation 164
Tab. 218. CryptoRequest AES CMAC - AES CMAC Sign Update Operation 164
Tab. 219. CryptoRequest AES CMAC - AES CMAC Sign Finalize Operation 165
Tab. 220. CryptoRequest AES CMAC - AES CMAC Sign One-shot Operation 165
Tab. 221. CryptoRequest AES CMAC - Format of crypto API AES CMAC Sign response data .. 165
Tab. 222. CryptoRequest AES CMAC - AES CMAC Verify Init Operation 165
Tab. 223. CryptoRequest AES CMAC - AES CMAC Verify Update Operation 166
Tab. 224. CryptoRequest AES CMAC - AES CMAC Verify Finalize Operation 166
Tab. 225. CryptoRequest AES CMAC - AES CMAC Verify One-shot Operation 166
Tab. 226. CryptoRequest AES CMAC - Format of crypto API AES CMAC Verify response data 167
Tab. 227. Error Code Description - AES Operation 167
Tab. 228. CryptoRequest AES AEAD - AES AEAD Initialize Operation 167
Tab. 229. CryptoRequest AES AEAD - Format of crypto API AES AEAD Initialize operation response data 168
Tab. 230. CryptoRequest AES AEAD - AES AEAD Update Operation 168
Tab. 231. CryptoRequest AES AEAD - Format of crypto API AES AEAD Update operation response data 168
Tab. 232. CryptoRequest AES AEAD - AES AEAD Finalize Operation 169

Tab. 233. CryptoRequest AES AEAD - Format of crypto API AES AEAD finalize operation response data 169
Tab. 234. CryptoRequest AES AEAD - AES AEAD One-Shot Operation 169
Tab. 235. CryptoRequest AES AEAD - Format of crypto API AES AEAD One-shot operation response data 170
Tab. 236. Error Code Description - AES Operation 170
Tab. 237. CryptoRequest - Write Internal Buffer Operation 171
Tab. 238. Response description - Write Internal Buffer 171
Tab. 239. CryptoRequest HMAC - HMAC Operation 171
Tab. 240. Response description - HMAC Verify Operation 172
Tab. 241. Response description - HMAC Sign Operation 172
Tab. 242. Error Code Description - HMAC Operation 172
Tab. 243. CryptoRequest HKDF - HKDF Extract and Expand Operation 173
Tab. 244. CryptoRequest HKDF - HKDF Expand Operation 173
Tab. 245. Response description - HKDF Operation 174
Tab. 246. Error Code Description - HKDF Operation 174
Tab. 247. CryptoRequest Echo - Echo Operation 174
Tab. 248. Response description - Echo Operation 174
Tab. 249. ManageGPIO 175
Tab. 250. Command Description - ManageGPIO 175
Tab. 251. Response description - ManageGPIO [else] . 176
Tab. 252. Error code description - ManageGPIO 176
Tab. 253. ReadGPIO 176
Tab. 254. Command Description - ReadGPIO 177
Tab. 255. Response description - ReadGPIO 177
Tab. 256. Error code description - ReadGPIO 178
Tab. 257. Command summary - ISOSelectFile 178
Tab. 258. Command description - ISOSelectFile 178
Tab. 259. Response description - ISOSelectFile 179
Tab. 260. Error code description - ISOSelectFile 179
Tab. 261. Command summary - ISOReadBinary 180
Tab. 262. Command description - ISOReadBinary 180
Tab. 263. Response description - ISOReadBinary 181
Tab. 264. Error code description - ISOReadBinary 181
Tab. 265. Command summary - ISOUpdateBinary 182
Tab. 266. Command description - ISOUpdateBinary 182
Tab. 267. Response description - ISOUpdateBinary 183
Tab. 268. Error code description - ISOUpdateBinary 183
Tab. 269. Limiting values 184
Tab. 270. Recommended operating conditions 185
Tab. 271. Electrical DC characteristics of GPIO1/2 186
Tab. 272. Electrical DC characteristics of I2C 187
Tab. 273. Electrical characteristics of IC supply voltage VCC 188
Tab. 274. Authentication application timing 188
Tab. 275. Nonvolitile memory timing characteristics 188
Tab. 276. Electrical AC characteristics of SDA, SCL 189
Tab. 277. Abbreviations 193
Tab. 278. Revision history 199

Figures

Fig. 1. A30 solution block diagram 2
Fig. 2. A30 for the consumable authentication 3
Fig. 3. A30 solution block diagram 3
Fig. 4. Block diagram 5
Fig. 5. ISO/IEC 7816-4 command response pair 9
Fig. 6. Authentication State Diagram 12
Fig. 7. Session key generation for Secure Messaging 24
Fig. 8. Plain Communication Mode 28
Fig. 9. Secure Messaging: MAC Communication mode 29
Fig. 10. Secure Messaging: CommMode.Full 30
Fig. 11. Secure Dynamic Messaging for Reading example 39
Fig. 12. Access conditions example 41
Fig. 13. Conceptual View of Host Verification Public Keys 64
Fig. 14. Conceptual View of a Certificate Repository 65
Fig. 15. Certificate Chain Example 66
Fig. 16. Crypto API Transient Buffer Format 79
Fig. 17. Crypto API Static Buffer Format 79
Fig. 18. ISOGeneralAuthenticate command protocol 93
Fig. 19. ISOInternalAuthenticate command protocol 94
Fig. 20. AuthenticateEV2First command protocol 96
Fig. 21. AuthenticateEV2NonFirst command protocol 99
Fig. 22. ProcessSM command protocol 101
Fig. 23. Protocol ProcessSM_Apply 102
Fig. 24. Protocol ProcessSM_Remove 103
Fig. 25. FreeMem command protocol 105
Fig. 26. SetConfiguration command protocol 106
Fig. 27. GetConfiguration command protocol 108
Fig. 28. ActivateConfiguration command protocol 110
Fig. 29. GetVersion command protocol 111
Fig. 30. GetCardUID command protocol 114
Fig. 31. ChangeKey command protocol 115
Fig. 32. GetKeySettings command protocol 118
Fig. 33. GetKeyVersion command protocol 121
Fig. 34. ManageKeyPair command protocol 122
Fig. 35. ManageCARootKey command protocol 125
Fig. 36. ManageCertRepo command protocol 127
Fig. 37. ReadCertRepo command protocol 130
Fig. 38. CreateStdDataFile command protocol 132
Fig. 39. CreateCounterFile command protocol 134
Fig. 40. GetFileIDs command protocol 135
Fig. 41. GetISOFileIDs command protocol 137
Fig. 42. GetFileSettings command protocol 138
Fig. 43. GetFileCounters command protocol 141
Fig. 44. ChangeFileSettings command protocol 142
Fig. 45. ReadData command protocol 148
Fig. 46. WriteData command protocol 150
Fig. 47. IncrementCounterFile command protocol 151
Fig. 48. CryptoRequestcommand protocol 153
Fig. 49. ManageGPIO command protocol 175
Fig. 50. ReadGPIO command protocol 176
Fig. 51. ISOSelectFile command protocol 178
Fig. 52. ISOReadBinary command protocol 180
Fig. 53. ISOUpdateBinary command protocol 182
Fig. 54. Input characteristics of GPIO1/2 187
Fig. 55. Package outline WLCSP (Top view) 190
Fig. 56. Package outline WLCSP (Bottom view) 190
Fig. 57. Package outline HVQFN 191

Contents

1 General description 1
2 Features and use cases 2
2.1 Use cases 2
2.2 Key features 2
2.3 Configuration 2
2.4 Configuration as authenticator 3
2.5 Configuration to secure IoT applications 3
3 Ordering information 4
4 Block diagram 5
5 Pin description 6
6 Functional description 7
6.1 I2C support 7
6.1.1 I2C parameter values 7
6.1.1.1 Target address 7
6.1.1.2 Communication interface parameters 7
6.1.2 I2C Application Remarks 8
6.1.2.1 Power Management 8
6.1.2.2 Write after Write behavior 8
6.2 Command format and chaining 8
6.2.1 Native command format 8
6.2.2 ISO/IEC7816-4 communication frame 9
6.2.3 Command chaining 10
6.3 Authentication and Secure Messaging 10
6.3.1 Authentication overview 10
6.3.2 SIGMA-I authentication with ISOGeneralAuthenticate 13
6.3.2.1 Session keys 13
6.3.2.2 Message types 13
6.3.2.3 Protocol exchange – Host as initiator 15
6.3.2.4 Protocol exchange – Host as responder 16
6.3.2.5 SIGMA-I session key generation 17
6.3.2.6 A30 Signature generation 18
6.3.2.7 SIGMA-I: Verification of the host 19
6.3.3 ECC-based card-unilateral authentication 19
6.3.3.1 Data structures and notations 20
6.3.3.2 Cryptographic primitives 20
6.3.3.3 ISOInternalAuthenticate 20
6.3.3.4 Authentication overview 21
6.3.4 AES-based Symmetric Authentication 22
6.3.4.1 Command AuthenticateEV2First 22
6.3.4.2 Command AuthenticateEV2NonFirst 23
6.3.4.3 Session Key Generation 23
6.3.5 AuthenticationCounter and Limit 25
6.3.6 EV2/AES secure messaging 26
6.3.6.1 Transaction Identifier 26
6.3.6.2 Command Counter 26
6.3.6.3 MAC Calculation 27
6.3.6.4 Encryption 27
6.3.6.5 Session Key Generation 27
6.3.6.6 Communication Modes 28
6.3.6.7 Plain Communication Mode 28
6.3.6.8 MAC Communication Mode 28
6.3.6.9 Full Communication Mode 29
6.3.7 Controller Session Key Usage 30
6.3.7.1 ProcessSM 30
6.3.7.2 ProcessSM_Apply 31
6.3.7.3 ProcessSM_Remove 31
6.3.8 Secure Dynamic Messaging 31
6.3.8.1 SDM Read Counter 32
6.3.8.2 SDM Read Counter Limit 33
6.3.8.3 PICCData 33
6.3.8.4 Encryption of PICCData 34
6.3.8.5 GPIOStatus 35
6.3.8.6 SDMENCFileData 35
6.3.8.7 Encryption of SDMENCFileData 36
6.3.8.8 SDMMAC 36
6.3.8.9 MAC Calculation 37
6.3.8.10 SDMSIG 37
6.3.8.11 Signature Calculation 38
6.3.8.12 SDM Session Key Generation 38
6.3.8.13 Output Mapping Examples 39
6.4 Access Rights Management 39
6.4.1 Access conditions 39
6.4.2 CARootKey access rights 41
6.4.3 Certificate access rights 42
6.5 Card Memory and Configuration Management 43
6.5.1 Card Version 43
6.5.1.1 Command GetVersion 44
6.5.2 Card configuration 44
6.5.2.1 Command SetConfiguration 44
6.5.2.2 Command GetConfiguration 55
6.5.2.3 Memory management 55
6.6 Symmetric Key Management 56
6.6.1 Key Types 56
6.6.2 Key Versioning 56
6.6.3 Symmetric Keys 56
6.6.3.1 AppMasterKey 57
6.6.3.2 AppKey 57
6.6.3.3 SDMMetaReadKey 57
6.6.3.4 SDMFileReadKey 57
6.6.3.5 AppPrivacyKey 58
6.6.3.6 CryptoRequestKey 58
6.6.4 Key Management Commands 58
6.6.4.1 Command ChangeKey 58
6.6.4.2 Command GetKeySettings 59
6.6.4.3 Command GetKeyVersion 60
6.7 Asymmetric Key Management 60
6.7.1 ECCPrivateKey Management 60
6.7.1.1 Command ManageKeyPair 60
6.7.1.2 ECCPrivateKey Key Usage Limit 60
6.7.1.3 ECCPrivateKey Information Retrieval 61
6.7.2 CARootKey Management 62
6.7.2.1 Command ManageCARootKey 62
6.7.2.2 CARootKey Information Retrieval 62
6.7.3 PICC/MF level 62
6.7.3.1 ECCPrivateKey entries 62
6.7.4 Application/DF level 62
6.7.4.1 ECCPrivateKey entries 62
6.7.4.2 CARootKey entries 62
6.7.5 Memory Consumption 63
6.7.6 Certificate Cache 63

6.8 Certificate Management 64
6.8.1 ECC Certificate Repository Management 64
6.8.1.1 Create Certificate Repository 65
6.8.1.2 Load Public Key Certificate Chain 65
6.8.1.3 Certificate Mapping Table 66
6.8.1.4 Activate Certificate Repository 67
6.8.2 Read Certificate Repository 68
6.9 Application Management 68
6.9.1 Application Selection 68
6.9.2 Application Definition 68
6.10 File Management 68
6.10.1 File Types 68
6.10.1.1 FileType.StandardData 68
6.10.1.2 FileType.Counter 69
6.10.2 File Access Rights Management 69
6.10.2.1 Secure Dynamic Messaging Related Access Rights 70
6.10.2.2 Access right association with commands 71
6.10.2.3 Command ChangeFileSettings 72
6.10.3 File Information Retrieval 73
6.10.3.1 Command GetFileSettings 73
6.10.3.2 Command GetFileCounters 73
6.10.3.3 Command GetFileIDs 74
6.10.3.4 Command GetISOFileIDs 74
6.10.4 File Creation 75
6.10.4.1 Command CreateStdDataFile 75
6.10.4.2 Command CreateCounterFile 75
6.10.5 Memory Consumption 75
6.10.6 File Definition 76
6.11 Data Management 77
6.11.1 Standard Data Files 77
6.11.1.1 Command ReadData 77
6.11.1.2 Command WriteData 78
6.11.2 Counter Files 78
6.11.2.1 Command IncrementCounterFile 78
6.12 Crypto API 79
6.13 GPIO Management 81
6.13.1 Command ManageGPIO 81
6.13.2 Command ReadGPIO 81
6.13.3 Mirroring in the NDEF message 83
6.13.4 Authentication notification 83
6.14 Timer Support 83
6.14.1 Authority Watchdog Timers 83
6.14.2 Halt Watchdog Timer 84
6.15 ISO/IEC 7816-4 Support 84
6.15.1 Standard ISO/IEC 7816-4 commands 84
6.15.1.1 Byte order 84
6.15.1.2 Security concepts of standard ISO/IEC 7816-4 commands 84
6.15.1.3 Error Handling 85
6.15.1.4 ISOSelectFile 85
6.15.1.5 ISOReadBinary 86
6.15.1.6 ISOUpdateBinary 86
6.16 Trust Provisioning 87
6.16.1 Originality Check Key Pair and Certificate 87
6.16.1.1 Originality Key Pair 87
6.16.1.2 Originality Certificate 87
6.16.1.3 Card-unilateral authentication 88
6.16.2 Application Key Pair and Certificate 88
6.16.2.1 Application Key Pair 88
6.16.2.2 Application Certificate 88
6.16.3 Commercial customization options 89
6.17 Security 89
6.17.1 Introduction 89
6.17.2 Reset 90
6.17.3 Sensor Architecture 90
6.17.4 Scalable Security 90
7 Command set 91
7.1 Introduction 91
7.2 Supported commands and APDUs 91
7.3 Authentication and Secure Messaging 93
7.3.1 ISOGeneralAuthenticate 93
7.3.2 ISOInternalAuthenticate 94
7.3.3 AuthenticateEV2First 96
7.3.4 AuthenticateEV2NonFirst 99
7.3.5 ProcessSM 101
7.3.6 ProcessSM_Apply 102
7.3.7 ProcessSM_Remove 103
7.4 Memory and Configuration Management 105
7.4.1 FreeMem 105
7.4.2 SetConfiguration 106
7.4.3 GetConfiguration 108
7.4.4 ActivateConfiguration 110
7.4.5 GetVersion 111
7.4.6 GetCardUID 114
7.5 Symmetric Key management 115
7.5.1 ChangeKey 115
7.5.2 GetKeySettings 118
7.5.3 GetKeyVersion 121
7.6 Asymmetric Key Management 122
7.6.1 ManageKeyPair 122
7.6.2 ManageCARootKey 125
7.6.3 GetKeySettings 127
7.7 Certificate Management 127
7.7.1 ManageCertRepo 127
7.7.2 ReadCertRepo 130
7.8 File Management 132
7.8.1 CreateStdDataFile 132
7.8.2 CreateCounterFile 134
7.8.3 GetFileIDs 135
7.8.4 GetISOFileIDs 137
7.8.5 GetFileSettings 138
7.8.6 GetFileCounters 141
7.8.7 ChangeFileSettings 142
7.9 Data Management 148
7.9.1 ReadData 148
7.9.2 WriteData 150
7.9.3 IncrementCounterFile 151
7.10 Crypto API 153
7.10.1 CryptoRequest SHA 154
7.10.2 CryptoRequest RNG 156
7.10.3 CryptoRequest ECC_Sign 156
7.10.4 CryptoRequest ECC_Verify 158
7.10.5 CryptoRequest ECC DH 160
7.10.6 CryptoRequest AES 162
7.10.7 CryptoRequest AES CMAC 164
7.10.8 CryptoRequest AES AEAD 167
7.10.9 CryptoRequest Write Internal Buffer 171

7.10.10 CryptoRequest HMAC 171
7.10.11 CryptoRequest HKDF 173
7.10.12 CryptoRequest Echo 174
7.11 GPIO Management 175
7.11.1 ManageGPIO 175
7.11.2 ReadGPIO 176
7.12 ISO7816-4 Support 178
7.12.1 ISOSelectFile 178
7.12.2 ISOReadBinary 180
7.12.3 ISOUpdateBinary 182
8 Limiting values 184
9 Recommended operating conditions 185
10 Characteristics 186
10.1 DC characteristics 186
10.1.1 General-purpose I/O interface 186
10.1.2 I2C interface 187
10.1.3 Power Consumption 188
10.2 AC characteristics 188
10.3 I2C Bus Timings 189
10.4 EMC/EMI 189
11 Package information 190
11.1 WLCSP 16 190
11.2 HVQFN 20 191
12 Abbreviations 193
13 References 196
14 Note about the source code in the document 198
15 Revision history 199
Legal information 200