GlobalPlatform Technology

Secure Channel Protocol '04'

Card Specification v2.3 – Amendment K

Version 1.0.1.10 (target 1.0.2)

Public Review

October 2024

Document Reference: GPC_SPE_182

Contents

Tables

Figures

1 INTRODUCTION

This document describes Secure Channel Protocol '04' (SCP04), where the building blocks of the cryptography protocol:

- Data Derivation
- Message Authentication Code
- Rekeying
- Cipher
- Sensitive Data Encryption
- Unpredictable Number Generation

…become configurable to allow for cryptographic agility of the protocol specification.

The protocol protects bidirectional communication between an Off-Card Entity and a Security Domain (decryption/MAC verification for incoming commands, encryption/MAC generation for Security Domain responses).

GlobalPlatform’s *Cryptographic Algorithm Recommendations* ([GPAR]) provides latest recommendations on cryptographic algorithms and protocols.

1.1 Audience

This amendment is intended primarily for card implementers and developers of on-card applications and off-card applications communicating with secure channel protocols.

It is assumed that the reader is familiar with smart cards and smart card production, and in particular familiar with the GlobalPlatform Card Specification ([GPCS]).

1.2 IPR Disclaimer

Attention is drawn to the possibility that some of the elements of this GlobalPlatform specification or other work product may be the subject of intellectual property rights (IPR) held by GlobalPlatform members or others. For additional information regarding any such IPR that have been brought to the attention of GlobalPlatform, please visit https://globalplatform.org/specifications/ip-disclaimers/. GlobalPlatform shall not be held responsible for identifying any or all such IPR, and takes no position concerning the possible existence or the evidence, validity, or scope of any such IPR.

1.3 References

Table 1-1: Normative References

Standard / Specification	Description	Ref
GlobalPlatform Card Specification GPC_SPE_034	GlobalPlatform Technology Card Specification v2.3.1	[GPCS]
GPCS Amendment A GPC_SPE_007	GlobalPlatform Technology Confidential Card Content Management – Card Specification v2.3 – Amendment A v1.2	[Amd A]

Standard / Specification	Description	Ref
GPCS Amendment D GPC_SPE_014	GlobalPlatform Technology Secure Channel Protocol '03' – Card Specification v2.3 – Amendment D v1.2	[Amd D]
GP Algorithm Recommendations GP_TEN_053	GlobalPlatform Technology Cryptographic Algorithm Recommendations v2.0 (or later)	[GPAR]
FIPS PUB 140-3	Federal Information Processing Standard 140-3, Security Requirements for Cryptographic Modules	[FIPS 140-3]
FIPS PUB 197	Federal Information Processing Standard 197, Advanced Encryption Standard (AES), November 2001	[FIPS 197]
GB/T 15852.1-2020	Information technology. Security techniques-Message authentication codes -- Part 1: Mechanisms using a block cipher	[SM4 MAC]
GB/T 17964-2021	Information technology. Security techniques-Message authentication codes -- Modes of operation for a block cipher	[SM4 Cipher]
ISO/IEC 8825-1	Information technology – ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)	[ISO 8825-1]
NIST SP 800-108 Revision 1	Recommendation for Key Derivation Using Pseudorandom Functions, October 2021	[NIST 800-108]
NIST SP 800-38A	Recommendation for Block Cipher Modes of Operation: Methods and Techniques, 2001	[NIST 800-38A]
NIST SP 800-38B	Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication, October 2016	[NIST 800-38B]
NIST SP 800-38D	Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC	[NIST 800-38D]
NIST SP 800-56A Revision 3	Recommendation for Pair-Wise Key-Establishment Schemes Using Discrete Logarithm Cryptography	[NIST 800-56A]
NIST SP 800-56B Revision 2	Recommendation for Pair-Wise Key Establishment Using Integer Factorization Cryptography	[NIST 800-56B]
NIST SP 800-56C Revision 2	Recommendation for Key-Derivation Methods in Key-Establishment Schemes	[NIST 800-56C]
NIST SP 800-57 Part 1 revised	Recommendation for Key Management – Part 1: General (Revised), May 2020	[NIST 800-57]
NIST SP 800-78-4	Cryptographic Algorithms and Key Sizes for Personal Identity Verification, May 2015	[NIST 800-78-4]
BSI TR-03111, Version 2.10	BSI Technical Guideline TR-03111: Elliptic Curve Cryptography	[TR 03111]

Standard / Specification	Description	Ref
ISO/IEC 18033-3:2010/AMD 1:2021	Information technology — Security techniques — Encryption algorithms — Part 3: Block ciphers — Amendment 1: SM4	[SM4]

1.4 Terminology and Definitions

Terms used in this document are defined in GlobalPlatform Card Specification ([GPCS]).

1.5 Abbreviations

Selected abbreviations and notations used in this document are included in Table 1-2. Additional abbreviations and notations are defined in [GPCS].

Table 1-2: Abbreviations

Abbreviation	Meaning
AEAD	Authenticated Encryption with Associated Data
AES	Advanced Encryption Standard
CASD	Controlling Authority Security Domain
CBC	Cipher Block Chaining
C-DECRYPTION	Command Decryption
CMAC	Cipher-based MAC (see note following table)
C-MAC	Command MAC (see note following table)
DGI	Data Grouping Identifier
FCI	File Control Information
ICV	Initial Chaining Vector
ISO	International Organization for Standardization
IV	Initialization Vector
KDF	Key Derivation Function
Key-DEK	Static Data Encryption Key
Key-ENC	Static Secure Channel Encryption Key
Key-MAC	Static Secure Channel Message Authentication Code Key
Lc	Exact length of command data in a case 3 or case 4 command
Le	Maximum length of data expected in response to a case 2 or case 4 command
LV	Length Value
MAC	Message Authentication Code
OCE	Off-Card Entity
PRF	Pseudorandom Function

Abbreviation	Meaning
R-ENCRYPTION	Response Encryption
R-MAC	Response MAC
SCP	Secure Channel Protocol
SD	Security Domain
S-ENC	Secure Channel command and response encryption session key
SM4	SM4 block cipher algorithm (Chinese)
S-MAC	Secure Channel C-MAC session key
S-RMAC	Secure Channel R-MAC session key
TLV	Tag Length Value

Note: C-MAC is the abbreviation used in [GPCS] for the MAC appended to command APDUs. This is not to be confused with CMAC, which is the abbreviation for a MAC calculation scheme specified in [NIST 800-38B].

1.6 Revision History

GlobalPlatform technical documents numbered *n*.0 are major releases. Those numbered *n*.1, *n*.2, etc., are minor releases where changes typically introduce supplementary items that do not impact backward compatibility or interoperability of the specifications. Those numbered *n.n*.1, *n.n*.2, etc., are maintenance releases that incorporate errata and clarifications; all non-trivial changes are indicated, often with revision marks.

Table 1-3: Revision History

Date	Version	Description
July 2023	v1.0	Public Release
Jan 2024	v1.0.1	Public Release showing all non-trivial changes since v1.0. This version adds several clarifications, in particular (but not only) about Data Derivation algorithms and their usage, and the generation of card and host challenges And cryptograms.
Oct 2024	v1.0.1.10	Public Review
TBD	v1.0.2	Public Release showing all non-trivial changes since v1.0. • There was an ambiguity of the term "Protocol Identifier" in v1.0. The new term "Protocol Configuration Identifier" is introduced to solve this ambiguity. • This version clarifies the supported Security Levels of SCP04 (e.g. clarifies that R-MAC is only supported in combination with C-MAC). • In version 1.0, section 7.4 explained that SCP04 keys may be associated with Protocol Configuration Identifier(s). But no data structure was provided to allow for this association. This version defines a mechanism to associate an SCP04 key set with an SCP identifier and Protocol Configuration Identifier(s). See sections 7.4 and 7.1. • The DGI/tag of the Protocol Configuration List is changed from 'BF71' to '9F71'. This data object contains primitive data rather than constructed data. • The Protocol Configuration List is now returned in the INITIALIZE UPDATE response data field. This simplifies the use of the Protocol Configuration List during session key derivation at the off-card side. • Several clarifications about command handling (especially when commands are rejected) are added.

2 SECURE CHANNEL PROTOCOL '04'

2.1 Scope of the Document

This document specifies a secure channel protocol, named Secure Channel Protocol '04' (SCP04), based on symmetric cryptography for mutual authentication, secure channel initiation, and secure messaging.

SCP04 describes a generic protocol as a combination of Protocol Functions, each of which is configured to use a specific cryptographic algorithm, e.g. AES, SHA-256.

2.2 Use Cases and Requirements

SCP04 is intended to ensure the integrity and confidentiality of the communication between a Secure Element and an Off-Card Entity for the following use cases:

- Card Content Management
- Application-specific personalization
- Secure Communication required by applications

Protocol Functions are specified in an abstract manner so that new algorithms providing the required functionality may be added in future. Every Protocol Function is instantiated with a specific algorithm (see section 3.2 for examples). The Protocol Functions shall be combined into a Protocol Configuration as specified in section 3.3.

The secure channel can be embedded into complex use cases, e.g. installation of payment credentials on cards or wearables, production systems, and remote provisioning of cell phone subscriptions. These use cases are out of scope of this document.

3 PROTOCOL FUNCTIONS AND ALGORITHMS

SCP04 is composed of a set of Protocol Functions (see section 3.1). Each Protocol Function is implemented by a cryptographic algorithm; some of the choices for these are described in section 3.2. A particular set of such algorithms represents a particular implementation of the SCP04 protocol and is known as a Protocol Configuration. Protocol Configurations, consisting of complete and consistent sets of algorithms for Protocol Functions, are defined in section 3.3.

Each Protocol Configuration is assigned a unique Protocol Configuration Identifier, which is used to indicate to the Security Domain the protocol that the OCE wishes to use for a specific secure channel. A Security Domain may support several such configurations, in which case the configuration to be used may be negotiated at the beginning of the session. In order to support this negotiation, each Security Domain shall contain a Protocol Configuration List, the list of protocols that it supports, identified by their Protocol Configuration Identifiers. This list can be obtained by OCEs using the GET DATA command (see section 7.1).

3.1 Protocol Functions

Each Protocol Function declares a list of input and output data to allow replacement of an algorithm in a Protocol Configuration.

Note that some of the algorithms may not require all of the inputs.

3.1.1 Data Derivation

The Data Derivation ~~algorithm~~ Protocol Function is used to generate keys, ~~pseudo-random~~ predictable card challenges, and cryptograms. ~~The Data Derivation Protocol Function~~ It accepts the following inputs:

- ~~Amount~~ Length of data to be generated
- Derivation constant
- Input data (the context data)
- Derivation Key ~~(for key-based derivation functions)~~

It generates the following output:

- Derived data

All Data Derivation schemes use the following derivation constants:

Table 3-1: Derivation Constants

b8	b7	b6	b5	b4	b3	b2	b1	Description
0	0	0	0	0	0	0	x	Authentication cryptogram generation
0	0	0	0	0	0	0	0	– card cryptogram
0	0	0	0	0	0	0	1	– host cryptogram
0	0	0	0	0	0	1	0	Card challenge generation
0	0	0	0	0	1	x	x	Key derivation
0	0	0	0	0	1	0	0	– derivation of S-ENC
0	0	0	0	0	1	1	0	– derivation of S-MAC
0	0	0	0	0	1	1	1	– derivation of S-RMAC
all other values								RFU

3.1.2 Message Authentication Code

The MAC Protocol Function accepts the following input:

- MAC key
- MAC data (algorithm specific, e.g. MAC Chaining Value for CMAC, IV for AEAD algorithms)
- Input data

It generates the following output:

- C-MAC or R-MAC
- Updated MAC Chaining Value / IV

3.1.3 Rekeying

The Rekeying Protocol Function updates S-ENC, S-MAC, and S-RMAC for each command. It accepts the following input:

- Counter or nonce byte string, may be ignored by the specific algorithm
- Input key

It generates the following output:

- Output key

3.1.4 Cipher

The Cipher Protocol Function accepts the following input:

- Key
- Initialization data (e.g. ICV/IV)
- Mode (encryption, decryption)
- Input data
- Associated data (present for AEAD only)

It generates the following output:

- Encrypted / decrypted data (including the authentication tag for AEAD encryption)
- Updated ICV/IV

On encryption, data is padded as required by the algorithm. On decryption, the padding is removed.

3.1.5 Sensitive Data Encryption

The Sensitive Data Encryption Protocol Function accepts the following input:

- Key
- Data
- Mode (encryption/decryption)

It generates the following output:

- Encrypted / Decrypted data

The Initial Chaining Vector (if needed) is defined by the algorithm.

On encryption, the input data is padded as required by the algorithm. The padding is removed on decryption.

3.1.6 Unpredictable Number Generation

The Unpredictable Number Generator Protocol Function accepts the following input:

- Length of requested random (in bytes)
- Seed data

It generates the following output:

- Random data

3.2 Algorithms

This section defines specific algorithms for Protocol Functions, together with their ~~Protocol~~ Identifiers. For each algorithm, the input parameters are those listed in section ~~3.1.1~~ 3.1.

The specification of algorithms presented in this section is not intended to be exhaustive or exclusive, and may be extended in later versions of the specification.

3.2.1 Data Derivation

3.2.1.1 CMAC-based Data Derivation Algorithm '10'

Data Derivation shall use KDF in counter mode as specified in [NIST 800-108]. The PRF used in the KDF shall be CMAC as specified in [NIST 800-38B]. The block cipher used in the PRF shall be the one corresponding to the type of the input key (e.g. AES or SM4) and shall have a block size of at least 128 bits.

Note that the algorithm specified here is the standard KDF in counter mode and is not the version described at the end of section 4.1 ~~in Appendix B~~ of [NIST 800-108], which counters a threat (described in Annex B of [NIST 800-108]) where the attacker already knows the secret key and uses this knowledge to control the output of the data derivation algorithm.

The ~~algorithm~~ KDF takes as input ~~fixed input data plus the iteration counter, which shall be the concatenation of~~ the following items in the given sequence (note that [NIST 800-108] allows the reordering of input data fields as long as the order, coding, and length of each field is unambiguously defined):

- A 12-byte label consisting of 11 bytes with value '00' followed by the 1-byte Derivation Constant (see Table 3-1).
- A 1-byte separation indicator with value '00'.
- A 2-byte integer “L” specifying the length in bits of the derived data. The length is given by the first input parameter.
- A 1-byte counter as specified in the KDF, which may take the values '01', '02', … according to the ~~amount~~ length of derived data (L) required (when the PRF has to be called repeatedly to generate enough derived data).
- Context data (composed of the input data specified for this protocol function; see section 3.1.1).

~~For the generation of session keys, the input data (the context parameter of the KDF) shall be set to the concatenation of the Host Challenge, the Card Challenge, the selected Protocol Identifier, and the Protocol Configuration List that was initially offered to the OCE.~~

~~The input data and the input key used for the generation of other data items are defined where relevant in the following sections.~~

3.2.1.2 Hash-based Data Derivation Algorithm '1x'

This algorithm implements data derivation using the “X9.63 Key Derivation Function” as defined in BSI TR 03111 ([TR 03111]). Data is derived from the repeated hashing of the concatenation of the shared secret, a counter, and the shared information ~~(the input data)~~.

The shared secret shall be the input Derivation Key (see section 3.1.1).

~~For key derivation, the shared information is~~ The shared information shall be the concatenation of the following values:

- The input Derivation Constant (see Table 3-1)
- The input data specified for this protocol function (see section 3.1.1)
- ~~Key usage qualifier (1 byte)~~
- ~~Key type (1 byte)~~
- ~~Key length (1 byte)~~
- ~~The Host Challenge~~
- ~~The Card Challenge~~
- ~~The selected Protocol Identifier~~
- ~~The Protocol Configuration List~~

~~With the exception of the Derivation Constant, these are provided in the input data block.~~

~~The algorithm may also be used to generate other data items, in which case the input data and the shared secret will be dependent on the data item being generated. This will be indicated where relevant in the following sections.~~

The hash function is defined by the value of 'x' in the algorithm identifier '1x':

x = 2: SHA-256

x = 3: SHA-384

x = 4: SHA-512

x = 5: SHA-3-256

x = 6: SM3

x = 7: AEAD algorithm in hashing mode, using the same AEAD algorithm as Cipher/MAC

x = 8: SHA-3-384

x = 9: SHA-3-512

3.2.2 Message Authentication Code

3.2.2.1 GlobalPlatform Core Specification MAC ~~Algorithm~~ '20'

This is CMAC with AES as specified in [GPCS] section B.2.2. The length of the generated CMAC is 16 bytes.

The length of the key used with this algorithm shall match the length of the key used by the cipher algorithm specified in the Protocol Configuration.

~~The following table lists the algorithm number and key length for this algorithm:~~

~~Algorithm number~~	~~'20'~~
~~Key length~~	~~Must match key length of encryption algorithm that is specified in the Protocol Configuration~~

3.2.2.2 SM4 MAC ~~Algorithm~~ '21'

This is SM4 MAC generation ([SM4 MAC]). The length of the generated MAC is 16 bytes.

~~The following table lists the algorithm number and key length for this algorithm:~~

~~Algorithm number~~	~~'21'~~
~~Key length~~	~~128 bits~~

3.2.2.3 AEAD MAC '42' ~~Algorithm~~

In the case of AEAD ciphers, the MAC is the authentication tag provided in the output of the AEAD cipher function (see section 3.2.4.3). The length of the generated MAC is defined by the AEAD cipher function.

If there are no data to be encrypted, the MAC is generated over the Associated Data only. If the security level indicates MAC only (command, response, or both), then all of the data forms the Associated Data.

In the Protocol Configuration, the algorithm number for the MAC Algorithm is set to the cipher algorithm number.

3.2.3 Rekeying

3.2.3.1 No Rekeying '30'

When this ~~option~~ algorithm is selected, no rekeying is performed.

Note: Rekeying is not specified in this version of the document.

3.2.4 Cipher

3.2.4.1 GPCS AES-CBC '40'

See [GPCS] section B.2.1 for the cipher and [GPCS] section B.2.3 for the padding.

This algorithm shall use an ICV of 16 bytes.

~~The following table lists the identification values and interface parameters for this algorithm:~~

~~**Algorithm number**~~	~~**'40'**~~
~~Length of ICV~~	~~16 bytes~~

3.2.4.2 SM4 '41'

This is SM4 in CBC encryption/decryption mode ([SM4 Cipher]).

This algorithm shall use an ICV of 16 bytes.

~~The following table lists the identification values and interface parameters for this algorithm:~~

~~**Algorithm number**~~	~~**'41'**~~
~~Length of ICV~~	~~16 bytes~~

3.2.4.3 AEAD: AES-GCM '42'

This is AES in Galois Counter Mode~~. Note that it~~, which supports both combined MAC and data confidentiality, and MAC alone generation if all input data are identified as Associated Data.

This algorithm shall use an ICV of 16 bytes and shall generate a MAC of 16 bytes.

~~The following table lists the identification values and interface parameters for this algorithm:~~

~~**Algorithm number**~~	~~**'42'**~~
~~Length of IV~~	~~16 bytes~~

3.2.5 Sensitive Data Encryption

3.2.5.1 GPCS AES-CBC '50'

This is AES CBC encryption with a 16-byte ICV set to zero~~ shall be used in conjunction with the Key-DEK~~.

~~The following table lists the identification values and interface parameters for this algorithm:~~

~~**Algorithm number**~~	~~**'50'**~~
~~Key-DEK~~	~~Static Data Encryption Key~~

3.2.5.2 SM4-CBC '51'

This is SM4 CBC encryption with a 16-byte ICV set to zero~~ shall be used in conjunction with the Key-DEK~~.

~~The following table lists the identification values and interface parameters for this algorithm:~~

~~**Algorithm number**~~	~~**'51'**~~
~~Key-DEK~~	~~Static Data Encryption Key~~

3.2.6 Unpredictable Number Generation

The following algorithms numbers have been allocated for Unpredictable Number Generation algorithms.

3.2.6.1 True Random '60'

Use internal hardware-based secure random generator.

3.2.6.2 Pseudo-Random '61'

Use pseudo-random generator as specified in [NIST 800-108].

3.3 Protocol Configurations and Identifiers

A Protocol Configuration consists of a complete and consistent set of algorithms for Protocol Functions. A Security Domain may support several such configurations, in which case the configuration to be used may be negotiated at the beginning of the session, using the exchange of Protocol Configuration Identifiers.

Each Protocol Configuration Identifier is encoded on one or more bytes and uniquely identifies a Protocol Configuration. For each byte in the Protocol Configuration Identifier, b8 is used to indicate the presence of a following byte according to [ISO 8825-1]:

- If b8 = 1, then there is a following byte.
- If b8 = 0, then there are no more bytes.

3.3.1 Defined Protocol Configurations

The Protocol Configurations supported by this version of the specification are shown in Table 3-2.

Table 3-2: Protocol Configurations

Protocol Configuration Identifier	Data Derivation	MAC	Rekeying	Cipher	Sensitive Data Encryption	Unpredictable
'01'	'10'	'20'	'30'	'40' (AES-CBC)	'50'	'60'
'02'	'10'	'21'	'30'	'41' (SM4)	'51'	'60'
'03'	'10'	'42'	'30'	'42' (AES-GCM)	'50'	'60'

Note that currently none of the Protocol Configurations make use of the hash-based data derivation algorithm.

3.3.2 Protocol Configuration Identifier Ranges

Table 3-3 shows the ranges of Protocol Configuration Identifiers:

Table 3-3: Allocated Ranges of Protocol Configuration Identifiers

Range	Use
'00'	Not used
'01' – '03'	As defined in section 3.3.1
'04' – '6F'	Reserved for future use by GlobalPlatform
'70' – '7F'	Reserved for proprietary use
'8000' – 'FF7F'	Reserved for future use by GlobalPlatform
All other values reserved for future use by GlobalPlatform	

3.3.3 Protocol Configuration List

The Protocol Configuration List is the concatenation of Protocol Configuration Identifiers identifying the ~~protocols~~ protocol configurations that the Security Domain supports. Protocol Configurations and their associated Protocol Configuration Identifiers are defined in section 3.3.1.

The Security Domain shall ensure that an attempt to open a Secure Channel using a Protocol Configuration Identifier that is not in its Protocol Configuration List shall fail.

Security Domains shall be personalized with the Protocol Configuration List using the STORE DATA command (see section 7.5). Note that before Security Domains enter the PERSONALIZED (or SECURED for ISD) state, a Protocol Configuration List must be personalized.

3.3.4 Card Capability Information

The Card Capability Information is defined in [GPCS] section 7.4.1.4 and section H.4.

The coding of the list of supported keys for SCP04, when employing AES, follows the same coding for SCP03 AES keys and is given in [GPCS] Table H-7.

The ~~new~~ "Protocol Configuration List of supported Protocol Configuration Identifiers for SCP04" shall be used to describe the SCP04 Protocol Configurations supported by the implementation.

Table 3-4 defines amended and additional conditional contents for the "SCP Information" TLV described in [GPCS] Table H-6.

Table 3-4: SCP Information

Tag	Length	Data / Description	Presence
'A0'	Variable	SCP Information	
'80'	1	SCP type ('04')	…
'81'	Variable	List of supported options for that protocol (see section 4.1)	
'82'	Variable	Supported keys for **SCP04**	Conditional
'85'	Variable	Protocol Configuration List of supported Protocol Configuration Identifiers for SCP04	Conditional

The following table describes the coding of the 'Supported keys for SCP04' TLV, used to identify the supported key length for an algorithm that accepts keys of differing lengths (for example AES).

Table 3-5: Supported Keys for SCP04

b8	b7	b6	b5	b4	b3	b2	b1	Description
					-	-	1	128 bit key
					-	1	-	192 bit key
					1	-	-	256 bit key
x	x	x	x	x				RFU (0)

4 Secure Channel Protocol Usage

This section defines the use of Secure Channel Protocol '04' to establish a secure session between an OCE and a Security Domain. An Application may implement the same interaction with an OCE by delegating the processing of protocol-related APDU commands to its associated Security Domain (see section 4.7).

4.1 Secure Communication Configuration

The three levels of security supported are:

- Mutual authentication – The Security Domain and the OCE each prove that they have knowledge of the same secrets.
- Integrity and data origin authentication – Ensuring that data received came from an authenticated entity in the correct sequence and without modification.
- Confidentiality – Data transmitted between authenticated parties is not viewable by an unauthorized entity.

In SCP04, the “i” parameter is formed as a bitmap on one byte as follows:

Table 4-1: Values of Parameter “i”

b8	b7	b6	b5	b4	b3	b2	b1	Description
				X	X	X	X	RFU (set to 0)
			0					Unpredictable card challenge
			1					Predictable card challenge
	0	0						No R-MAC/R-ENCRYPTION support
	0	1						R-MAC support / no R-ENCRYPTION support
	1	1						R-MAC and R-ENCRYPTION support
X								Reserved

Note: “i” is a sub identifier within an object identifier, and bit b8 is reserved for use in the structure of the object identifier according to [ISO 8825-1].

4.2 Secure Channel Setup

Mutual authentication is achieved through the process of initiating a Secure Channel and provides assurance to both the Security Domain and the OCE that they are communicating with an authenticated entity. If any step in the mutual authentication process fails, the process shall be restarted, i.e. new challenges and Secure Channel session keys shall be generated.

The Secure Channel is explicitly initiated by the OCE using the INITIALIZE UPDATE command.

Note: Implicit Secure Channel Initiation as described in [GPCS] is not defined for SCP04.

Explicit Secure Channel initiation provides the OCE with the opportunity to select the Key Version Number of the keys to be used in the generation of session keys. It also allows the OCE to indicate the level of security required for the current Secure Channel (integrity and/or confidentiality for command and/or response data) and to apply this level of security to all subsequent messages exchanged between the Security Domain and the OCE until the end of the Secure Channel Session.

The Secure Channel is always initiated by the OCE providing the Host Challenge (random data unique to this Secure Channel Session) to the Security Domain in the INITIALIZE UPDATE command.

The Security Domain, on receipt of this challenge, generates its Card Challenge ~~(again random data unique to this Secure Channel Session)~~ (see section 6.1.1). Using the Host Challenge, the Card Challenge, and the static keys indicated by the Keyset Version Number provided by the OCE (if these are applicable to the chosen protocol), the Security Domain creates Secure Channel session keys (see section 5.2) and generates the Card Cryptogram (see section 6.1.3).

The Card Cryptogram, the Card Challenge, the Secure Channel Protocol Identifier, and ~~key derivation data~~ Key Diversification Data are returned to the OCE.

The OCE generates session keys and verifies the Card Cryptogram. If this is successful, then the OCE has authenticated the Security Domain.

The OCE creates the Host Cryptogram and sends it to the Security Domain using the EXTERNAL AUTHENTICATE command, setting the required security level and appending the C-MAC.

The Security Domain verifies the C-MAC associated with the command and verifies the Host Cryptogram. If this is successful, then the Security Domain has authenticated the OCE and mutual authentication is complete.

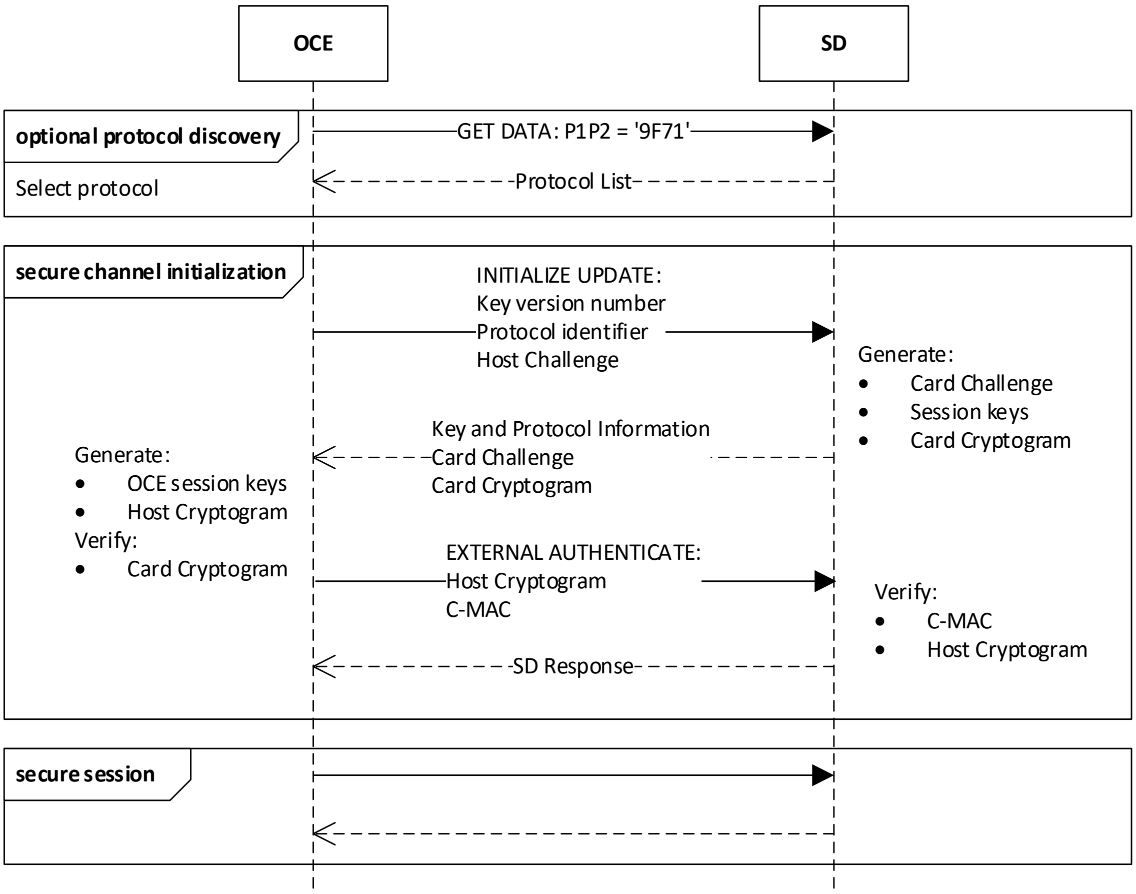
Prior to performing mutual authentication, the OCE may retrieve the Protocol List from the Security Domain using the GET DATA command. This is an optional step: The OCE might have alternative ways for determining the capability of the Security Domain (not least because of the keysets that it holds), but these are out of scope of this specification.

An SD supporting SCP04 shall implement the `SecureChannel` interface of the API specified in [GPCS]. An application associated with the SD may use this API to request the SD to handle SCP04 related commands.

The `processSecurity()` method shall handle the following APDU commands:

- INITIALIZE UPDATE
- EXTERNAL AUTHENTICATE
- GET DATA (without secure messaging) to get the Protocol List

Figure 4-1: Explicit Secure Channel Initiation Flow



Expanding the authentication process shown in the flow described in [GPCS] section 7.3.1, Security Domain Support for Secure Messaging, it can be seen how an application would use the services of a Security Domain to achieve the explicit Secure Channel initiation.

4.3 Session Key Negotiation

After receiving the INITIALIZE UPDATE command, the Security Domain shall:

- Extract the Host Challenge.
- Generate the Card Challenge (as specified in section 6.1.1).
- ~~Determine whether to use an unpredictable or predictable Card Challenge generation.~~
- ~~Increase the sequence counter if a predictable value is used, and generate the Card Challenge.~~
- Generate the Session Keys (as specified in section 5.2).
- ~~Apply the protocol's key derivation function (see section 3.1.1) to generate the session keys.~~
- Generate the Card Cryptogram (as specified in section 6.1.3).
- ~~Generate the Card Cryptogram over the rest of the response data.~~
- Return the following to the OCE:
  - Key diversification data
  - Key information
  - The sequence counter, if the predictable mechanism for Card Challenge generation is used
  - The Card Challenge
  - The Card Cryptogram

The OCE now has all the necessary information required to perform its own session key generation, verify the Card Cryptogram, and generate the contents of the EXTERNAL AUTHENTICATE command (Host Cryptogram and C-MAC).

After receiving the EXTERNAL AUTHENTICATE command, the Security Domain shall:

- Authenticate the OCE by verifying the C-MAC and the Host Cryptogram.
- Set the security level for the Secure Channel Session to the value of P1 provided in the command.

The session keys have now been validated by both parties and can be used to secure messages passed between them for the duration of the Secure Channel session.

4.4 Command APDU and Response APDU Processing

On receipt of the first command of the Secure Channel session (following mutual authentication), the Security Domain processes the following steps as required by the security level:

1. The Security Domain verifies the C-MAC of the incoming command by applying the MAC Protocol Function (see section 3.1.2) in conjunction with the S-MAC session key (generated during the mutual authentication process).
2. The Security Domain decrypts the command data by applying the Cipher Protocol Function (see section 3.1.4) with the S-ENC session key (generated during the mutual authentication process).
3. The Security Domain processes the command data:
   - If ~~the command requires this~~ required, the Security Domain applies the Sensitive Data Decryption Protocol Function (see section 3.1.5) with the static Key-DEK key (of the key set selected for the current Secure Channel session).
4. The Security Domain encrypts the response data (if any) by applying the Cipher Protocol Function (see section 3.1.4) with the S-ENC session key (generated during the mutual authentication process).
5. The Security Domain calculates the MAC of the response data by applying the MAC Protocol Function (see section 3.1.2) with the S-RMAC session key (generated during the mutual authentication process). Note that the response data might consist of only the Status Bytes.
6. The Security Domain updates the session keys by applying the Rekeying Protocol Function~~, if required by the security level~~ (see section 3.1.3).
7. The Security Domain updates the chaining parameters (MAC Chaining Value, IV, etc., as appropriate for the algorithms in use).

The response is returned to the OCE. These steps are repeated for each command received during the Secure Channel session.

Note that for an AEAD algorithm, steps 1 and 2 and steps 4 and 5 will be performed simultaneously and the key used will always be the S-ENC session key. See sections 6.9.2 and 6.9.3.

4.5 Message Integrity

The C-MAC is generated using the S-MAC generated during the mutual authentication process, by applying the Message Authentication Code Protocol Function across the header and data field of an APDU command. Note that ~~the session key used to generate the MAC is dependent on the algorithm (~~for an AEAD algorithm, ~~will use~~ the S-ENC session key shall be used instead~~, not S-MAC;~~ (see section 6.9.5).

The Security Domain, on receipt of the message containing a C-MAC, using the same Secure Channel session key, performs the same operation and by comparing its internally generated C-MAC with the C-MAC received from the OCE is assured of the integrity of the full command.

If message data confidentiality has also been applied to the message, the C-MAC applies to the message data field after encryption has been performed. For AEAD algorithms it is calculated as part of the decryption process (see section 6.9).

The integrity of the sequence of commands being transmitted to the Security Domain is achieved by using the C-MAC of a command as part of the input for the computation of the C-MAC of the next command. At any point in time, the last C-MAC computed is part of the channel state and is referred to as the “MAC Chaining Value” further in this document. The first MAC Chaining Value is set to '00' (see section 6.2). This chaining (see Figure 6-5) ensures that commands have been received in the correct sequence.

The length of the C-MAC (and therefore the MAC Chaining Value) depends on the MAC algorithm. See section 3.2.2 for details.

The integrity of the response is chained to the command sequence integrity by using the MAC Chaining Value as input for the computation of the R-MAC on responses (see Figure 6-5).

Note: AEAD cipher functions use a different mechanism, whereby the MAC is used as the IV for the next invocation of the AEAD function; see section 6.8.

4.6 Message Data Confidentiality

The message data field is encrypted using the Cipher Protocol Function (see section 3.1.4), using the S-ENC session key generated during the mutual authentication process across the entire data field of the command message to be transmitted to the Security Domain, and if required also across the response transmitted from the Security Domain, regardless of its contents (clear text data and/or already protected sensitive data).

4.7 API and Security Level

A Security Domain implementing SCP04 shall implement the `SecureChannel` interface of the API specified in GlobalPlatform Card Specification ([GPCS]).

The `processSecurity()` method shall handle the following APDU commands:

- INITIALIZE UPDATE
- EXTERNAL AUTHENTICATE
- GET DATA (without secure messaging), to retrieve data objects specific to the SCP04 protocol (see section 7.1). **Note:** The implementation may allow retrieving other SD data objects.
- GET RESPONSE (without secure messaging), to retrieve GET DATA responses longer than 256 bytes (see section 7.1)
  - If the requested data object cannot be found or the SD does not allow it to be retrieved in this way, then the `processSecurity()` method shall throw an `ISOException` with reason code '6A88'. In this case, after catching and inspecting this exception, the calling application may decide to further process the GET DATA command. **Note:** This behavior allows the calling application to also implement support for the GET DATA command for the retrieval of application-specific data objects.
  - If the response to GET DATA or GET RESPONSE (see below) is 256 bytes or less, then the `processSecurity()` method shall behave as usual, i.e. it shall return the length of the response data present in the APDU buffer, and the calling application is responsible for sending such response data out (using the APDU object).
  - If the response to GET DATA or GET RESPONSE (see below) is longer than 256 bytes, the `processSecurity()` method shall write the next 256 bytes of response data in the APDU buffer and then throw an `ISOException` with reason code '61xx' with xx specifying the length of remaining response data ('00' if 256 bytes or more). In this case, the calling application is expected to catch the `ISOException` and inspect its reason code, and if that reason code is '61xx', then it shall assume 256 bytes of response data are present in the APDU buffer, send such response data out (using the APDU object) and then re-throw the `ISOException`.
- GET RESPONSE (without secure messaging), to retrieve remaining GET DATA response bytes.
  - The `processSecurity()` method shall implement the GET RESPONSE state machine allowing to retrieve remaining data. This state machine shall be reset if a new recognized GET DATA or any recognized protocol command is processed on the same logical channel. It shall not be reset by a call to the `processSecurity()` method performed on another logical channel (on that same SD): The implementation may either reject the call or, if enough resources are available, manage it independently.

The following shall apply for the Security Level:

- The Current Security Level of a communication not included in a Secure Channel Session shall be set to NO_SECURITY_LEVEL.
- For Secure Channel Protocol '04', the Current Security Level established in a Secure Channel Session is a bitmap combination of the following values: AUTHENTICATED, C_MAC, R_MAC, C_DECRYPTION, and R_ENCRYPTION.

The Current Security Level shall be set as follows:

- NO_SECURITY_LEVEL when a Secure Channel Session is terminated or not yet fully initiated.
- AUTHENTICATED after successful processing of the EXTERNAL AUTHENTICATE command; AUTHENTICATED shall be cleared once the Secure Channel Session is terminated.
- C_MAC after successful processing of the EXTERNAL AUTHENTICATE command with P1 indicating C-MAC (P1 = 'x1' or 'x3'; see section 7.3.3); C_MAC shall be cleared once the Secure Channel Session is terminated. Note that C_MAC is always combined with AUTHENTICATED and simultaneously set and cleared.
- C_DECRYPTION after successful processing of the EXTERNAL AUTHENTICATE command with P1 indicating Command Encryption (P1= 'x3'); C_DECRYPTION shall be cleared once the Secure Channel Session is terminated. Note that C_DECRYPTION is always combined with AUTHENTICATED and C_MAC and simultaneously set and cleared.
- R_MAC after successful processing of the EXTERNAL AUTHENTICATE command with P1 indicating R-MAC (P1='11~~x~~', '13', or ~~'3x'~~'33'; see section 7.3.3); R_MAC shall be cleared once the Secure Channel Session is terminated. Note that R_MAC is always combined with AUTHENTICATED and C_MAC and simultaneously set and cleared. R_MAC may also be combined with ~~C_MAC or~~ C_DECRYPTION (according to the P1 value of the EXTERNAL AUTHENTICATE command) and simultaneously set and cleared.
- R_ENCRYPTION after successful processing of the EXTERNAL AUTHENTICATE command with P1 indicating Response Encryption (P1= '33~~x~~'; see section 7.3.3); R_ENCRYPTION shall be cleared once the Secure Channel Session is terminated. Note that R_ENCRYPTION is always combined with AUTHENTICATED~~ and~~, R_MAC, and C_DECRYPTION and simultaneously set and cleared.

4.7.1 AEAD Security Levels

AEAD algorithms provide an efficient coupling of encryption and authentication. These processes can be performed on command data and response data indicated by the following security levels:

- C-MAC and C-DECRYPTION ('03')
- C-MAC, C-DECRYPTION, R-MAC, and R-ENCRYPTION ('33')

AEAD algorithms support the generation and verification of MACs without performing data encryption/decryption by setting the length of the plaintext data to zero, and the length of the associated data to be the length of the APDU. An example of such an algorithm is AES-GMAC, which is the special case of AES-GCM where the input plaintext is empty (see [NIST 800-38D]). MAC generation/verification without data encryption results in the following security levels being supported:

- C-MAC ('01')
- C-MAC and R-MAC ('11')
- C-MAC, C-DECRYPTION, and R-MAC ('13')

Note that MAC generation without data encryption is used in the construction of the EXTERNAL AUTHENTICATE command.

4.8 Protocol Rules

In accordance with the general rules described in [GPCS] section 10, the following protocol rules apply to Secure Channel Protocol '04':

- The successful initiation of a Secure Channel Session shall set the Current Security Level to the security level indicated in the EXTERNAL AUTHENTICATE command: It is at least set to AUTHENTICATED.
- The Current Security Level shall apply to the entire Secure Channel Session unless successfully modified at the request of the Application.
- When the Current Security Level is set to NO_SECURITY_LEVEL:
    - If the Secure Channel Session was aborted during the same Application Session, the incoming command shall be rejected with a security error.
    - Otherwise no security verification of the incoming command shall be performed. The Application processing the command is responsible to apply its own security rules.
- If a Secure Channel Session is active (i.e. Current Security Level at least set to AUTHENTICATED), the security of the incoming command shall be checked according to the Current Security Level regardless of the command secure messaging indicator:
    - When the security of the command is less than the Current Security Level, the command shall be rejected with a security error, the Secure Channel Session aborted, and the Current Security Level reset to NO_SECURITY_LEVEL.
    - If a security error is found, the command shall be rejected with a security error, the Secure Channel Session aborted, and the Current Security Level reset to NO_SECURITY_LEVEL.
    - In all other cases, the Secure Channel Session shall remain active and the Current Security Level unmodified. The Application is responsible for further processing the command.
- If a Secure Channel Session is aborted, it is still considered not terminated.
- The current Secure Channel Session shall be terminated (if aborted or still open) and the Current Security Level reset to NO_SECURITY_LEVEL in any of the following situations:
    - Attempt to initiate a new Secure Channel Session (new INITIALIZE UPDATE command)
    - Termination of the Application Session (e.g. selection of a new Application)
    - Termination of the associated logical channel
    - Termination of the Card Session (card reset or power off)
    - Explicit termination by the Application (e.g. invoking GlobalPlatform API)

5 CRYPTOGRAPHIC KEYS

5.1 Secure Channel Keys

Table 5-1: Secure Channel Keys

Key	Usage	Presence
Static Secure Channel Encryption Key (Key-ENC)	Generate session key for Decryption/Encryption	Mandatory
Static Secure Channel Message Authentication Code Key (Key-MAC)	Generate session key for Secure Channel authentication and Secure Channel MAC verification/generation	Mandatory
Static Data Encryption Key (Key-DEK)	Sensitive Data Decryption	Mandatory
Session Secure Channel Encryption Key (S-ENC)	Used for data confidentiality	Dynamic
Session Secure Channel Message Authentication Code Key for Command (S-MAC)	Used for data and protocol integrity	Dynamic
Session Secure Channel Message Authentication Code Key for Response (S-RMAC)	User for data and protocol integrity	Dynamic and Conditional

The Key Type parameter of the Control Reference Template present in a STORE DATA command when loading Secure Channel ~~Static Keys~~ keys into a Security Domain, shall take values listed in [GPCS] section 11.1.8, with additional clarifications provided below:

- Key Type '88' identifies AES keys used by AES-based algorithms (including AES-GCM).
- Key Type '89' identifies SM4 keys used by SM4-based algorithms.

A Security Domain supporting Secure Channel Protocol '04', including the Issuer Security Domain, shall have at least one complete key set containing a Key-ENC, a Key-MAC, and a Key-DEK key having the same length. When loaded to the Security Domain, these keys should be encrypted with a key of the same or higher strength. The card issuer may require that this recommendation be enforced depending on its security policy.

5.2 Session Key Derivation

Session keys shall be generated every time a Secure Channel is initiated and are used in the mutual authentication process. These session keys are then used for subsequent commands if the Current Security Level indicates that secure messaging is required. Session keys are generated to ensure that a different set of keys is used for each Secure Channel Session.

The session keys are derived from the static Secure Channel keys. The encryption key S-ENC is derived from Key-ENC. The Secure Channel MAC key S-MAC is derived from Key-MAC. Optionally (if the "i" parameter indicates R-MAC support), the Secure Channel R-MAC key S-RMAC is derived from Key-MAC. These session keys shall have the same length as the static key from which they are derived.

Additional notes for AEAD session keys:

- S-ENC: AEAD algorithms only use a single key for encryption/decryption and MAC generation, S-ENC. The authentication tag provides the MAC.
- S-MAC: This is only used to generate the Card/Host Cryptograms used in the mutual authentication mechanism (see sections 6.1.3 and 6.1.4). It is not used during the rest of the session.
- There is no requirement for a separate S-RMAC key.
- The use of session keys satisfies the restrictions on secure operation that are required for AEAD algorithms that reuse the encryption key (for example AES-GCM; see [NIST 800-38D] section 8.3).

~~Note that t~~The Key-DEK is not used to derive a session key. Instead of using session keys, sensitive data are encrypted using the static ~~data~~ Key-DEK, allowing for pre-encryption outside a Secure Channel Session.

~~Key derivation shall use the Data Derivation scheme defined in section 3.2.1, with Key Derivation Elements as indicated in the following table.~~

~~Table 5-2: Key Derivation Elements~~

~~Derived Session Key~~	~~Key Used in PRF~~	~~Derivation Constant (as defined in Table 3-1)~~
~~S-ENC~~	~~Key-ENC~~	~~'04'~~
~~S-MAC~~	~~Key-MAC~~	~~'06'~~
~~S-RMAC~~	~~Key-MAC~~	~~'07'~~

Key derivation shall use the Data Derivation algorithm selected by the protocol configuration. Following the definition of the Protocol Function given in section 3.1.1, the inputs to the Data Derivation process are:

- The ~~amount~~ length of key data ~~required~~ to be generated
- The Derivation Constant (see Table 3-1)
- The input data, ~~– Set~~ set to the concatenation of the Host Challenge, the Card Challenge, the selected Protocol Configuration Identifier, and the supported Protocol Configuration List
- The Derivation ~~key~~ Key ~~to be used in the PRF~~

~~The presence of random data is dependent on the key derivation algorithm.~~

~~5.2.1 AES Session Keys~~

~~AES session key derivation shall use the Key Derivation scheme defined in section 3.2.1.1, with Key Derivation Elements as indicated in section 5.2.~~

~~The length of the session keys shall be reflected in the parameter "L" (i.e. '0080' for AES-128 keys, '00C0' for AES-192 keys, and '0100' for AES-256 keys).~~

~~5.2.2 SM4 Session Keys~~

~~SM4 session key derivation shall use the Key Derivation scheme defined in section 3.2.1.1, with Key Derivation Elements as indicated in section 5.2.~~

~~SM4 keys are all 128 bits in length and this shall be reflected in the parameter "L" = '0080'.~~

~~5.2.3 AEAD Session Keys~~

~~Each AEAD keyset will consist of three keys Key-ENC, Key-MAC, and Key-DEK. Session keys are derived from the first two as part of the standard mutual authentication mechanism using the key derivation process defined for the algorithm. The session keys are used as follows:~~

- ~~S-ENC: AEAD algorithms only use a single key for encryption/decryption and MAC generation, S-ENC. The authentication tag provides the MAC.~~
- ~~S-MAC: This is only used to generate the Card/Host Cryptograms used in the mutual authentication mechanism (see sections 6.1.2 and 6.1.3). It is not used during the rest of the session.~~
- ~~Note that the use of session keys satisfies the restrictions on secure operation that are required for AEAD algorithms that reuse the encryption key (for example AES-GCM; see [NIST 800-38D] section 8.3).~~
- ~~Note also that there is no requirement for a separate R-MAC key.~~

6 CRYPTOGRAPHIC USAGE

6.1 Challenges and Authentication Cryptograms

Both the Security Domain and the OCE generate a challenge and an authentication cryptogram.

- The Host Challenge and Card Challenge~~s~~ are involved in the computation of the Host Cryptogram and Card Cryptogram, as well as in the generation of the session keys. ~~combined to provide the context for session key generation.~~
- The OCE verifies the Card Cryptogram, and the Security Domain verifies the Host Cryptogram.
- The Host Challenge, Card Challenge, Host Cryptogram, and Card Cryptogram all share the same length.

~~Note the following:~~

- ~~The length of the Host and Card Challenges shall be same and, while, they may be dependent on the protocol in use, must provide at least 128 bits of entropy (no less than 16 bytes in length).~~
- ~~The cryptogram lengths shall be the same as the length of the challenges.~~

6.1.1 Card Challenge

The Card Challenge shall either be predictable or unpredictable as indicated in the "i" parameter (see Table 4-1). If an unpredictable value is to be generated, it shall be calculated using the Unpredictable Number Generation algorithm indicated ~~in~~ by the selected Protocol Configuration. The length of the generated Card Challenge shall be the same as the length of the MAC applicable for the selected Protocol Configuration.

If the "i" parameter is configured for predictable challenge generation, the Card Challenge shall be calculated as follows:

- For each SCP04 key set, the Security Domain shall have a sequence counter of 3 bytes length. Whenever a key set is created or the whole key set is replaced by a single STORE DATA command, the sequence counter shall be set to zero.
- Whenever the generation of a predictable challenge is triggered by an INITIALIZE UPDATE command, the sequence counter shall be incremented and the new value shall be used in the calculation described below. When the maximum value is reached, the INITIALIZE UPDATE command shall be rejected with "Conditions of use not satisfied".
- The Card Challenge ~~is~~ shall be calculated using the Data Derivation algorithm of the selected Protocol Configuration ~~the Data Derivation Protocol Function (see sections 3.1.1 and 3.2.1)~~ with the ~~Static Key~~ static key Key-ENC of the selected key set and the Derivation Constant set to "card challenge generation" (i.e. '02'). The length of the challenge shall be reflected in the parameter "L" ~~(i.e. '0080')~~. The input data shall be set to the concatenation of the sequence counter (3 bytes) and the AID of the application invoking the `SecureChannel` interface (5 to 16 bytes).
- When used in conjunction with AEAD algorithms, the ~~Static Key~~ static key Key-MAC is used in the calculation of the Card Challenge, not Key-ENC.

6.1.2 Host Challenge

The length of the Host Challenge generated by the OCE shall be the same as the length of the MAC applicable for the selected Protocol Configuration.

~~6.1.2~~6.1.3 Card Authentication Cryptogram

The Card Cryptogram shall be calculated using the Data Derivation Protocol Function (see sections 3.1.1 and 3.2.1) of the Protocol Configuration selected by the OCE in the INITIALIZE UPDATE command. The input data shall be the concatenation of the host challenge and the card challenge; the key shall be the ~~Session~~ S-MAC Key; the Derivation Constant shall be set to “card cryptogram”. The length of the generated Card Cryptogram shall be the same as the length of the MAC applicable for the selected Protocol Configuration.

~~6.1.3~~6.1.4 Host Authentication Cryptogram

The Host Cryptogram shall be calculated using the Data Derivation Protocol Function (see sections 3.1.1 and 3.2.1) of the Protocol Configuration as selected by the OCE in the INITIALIZE UPDATE command. The input data shall be the concatenation of the host challenge and the card challenge; the key shall be the S-MAC Key; the Derivation Constant shall be set to “host cryptogram”. The length of the generated Host Cryptogram shall be the same as the length of the MAC applicable for the selected Protocol Configuration.

6.2 Message Integrity Using Explicit Secure Channel Initiation

SCP04 mandates the use of a MAC attached to the EXTERNAL AUTHENTICATE command. The Message Authentication Code Protocol Function (see sections 3.1.2 and 3.2.2) shall be used to generate the MAC.

For MAC verification, the MAC Chaining Value is set to bytes of '00'. In the case of AEAD processing the IV for the Cipher function is set to the Card Challenge.

Once the cryptogram is successfully verified, the C-MAC of the previous command becomes the MAC Chaining Value for the subsequent C-MAC verification / R-MAC generation. See section 6.7.

In the case of AEAD processing, the IV for the next invocation of the Cipher function is set according to section 6.8.

6.3 APDU Command C-MAC Generation and Verification

A C-MAC is generated by an OCE using the MAC algorithm (see sections 3.1.2 and 3.2.2) in the following manner:

- The key is the S-MAC session key.
- The MAC Chaining Value is as described in section 6.7.
- The input data is the full APDU command being transmitted to the Security Domain including the header (5 bytes) and the data field in the command message.
- Modification of the APDU command header ~~and padding~~ is required prior to the MAC ~~operation~~ algorithm being ~~performed~~ applied.

The MAC length is determined by the Message Authentication Code Protocol Function (see sections 3.1.2 and 3.2.2).

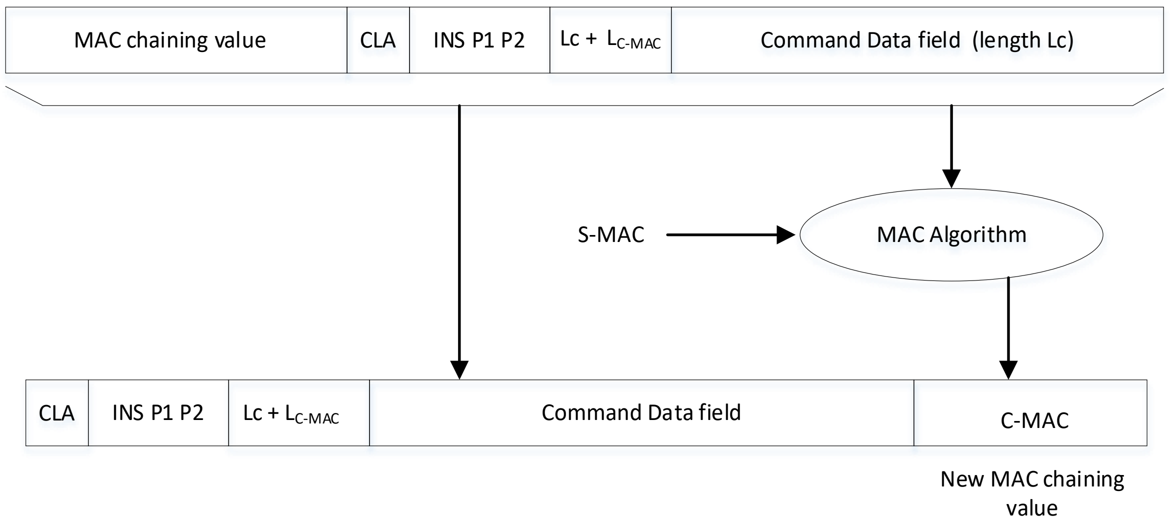
The rules for APDU command header modification are as follows:

- The length of the command message (Lc) shall be incremented by the length of the C-MAC to indicate the inclusion of the C-MAC in the data field of the command message.
- The class byte shall be modified for the generation or verification of the C-MAC: The logical channel number shall be set to zero, bit 4 shall be set to 0, and bit 3 shall be set to 1 to indicate GlobalPlatform proprietary secure messaging.
- If the Secure Channel Session is occurring on a Supplementary Logical Channel, the class byte shall be modified after the C-MAC generation to indicate the logical channel number.
- If logical channel number 4 to 19 is used, the GlobalPlatform proprietary secure messaging is indicated by setting bit 6 to 1 – see [GPCS] Table 11-11 and Table 11-12.
- Conversely, the logical channel number is discarded and, if required, the secure messaging indication is adjusted for the verification.
- The logical channel number is not part of the integrity protection by the channel because it is established independently and out of scope of the Secure Channel establishment.

The MAC generation mechanism is shown in Figure 6-1.

MAC verification is performed by repeating the generation process on the input APDU data and comparing the generated MAC with the one attached to the command data. If verification is successful then the MAC is detached from the command data and Lc is adjusted accordingly.

Figure 6-1: APDU C-MAC Generation



Note that if padding is required for the MAC generation, it is considered to be part of the MAC algorithm.

6.4 APDU Response R-MAC Generation and Verification

No R-MAC shall be generated and no protection shall be applied to a response that includes an error status word: In this case only the status word shall be returned in the response. All status words except '9000' and warning status words (i.e. '62xx' and '63xx') shall be interpreted as error status words.

The EXTERNAL AUTHENTICATE response does not include an R-MAC.

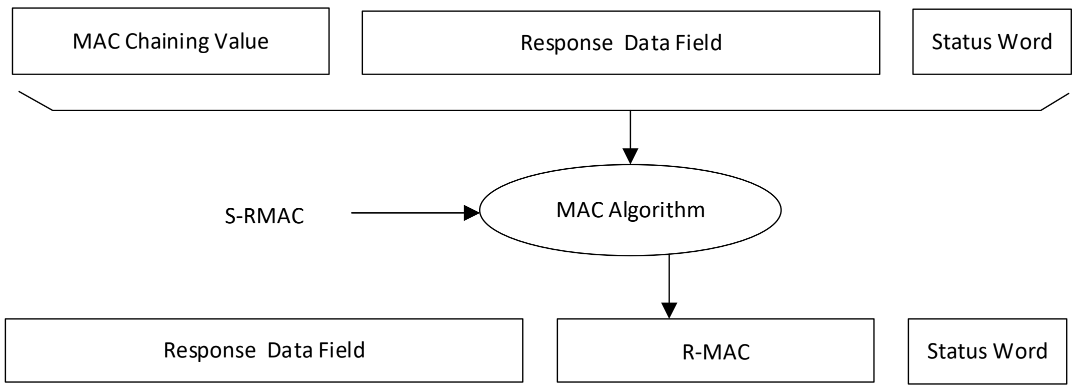
The R-MAC is computed using the MAC algorithm (see sections 3.1.2 and 3.2.2) with the following inputs:

- The key is the S-RMAC session key.
- The MAC Chaining Value is as described in section 6.7.
- The response data field (if present) and the status bytes.

The computed R-MAC becomes part of the response message, following the original response data and preceding the status bytes.

The R-MAC calculation is illustrated in Figure 6-2.

Figure 6-2: APDU R-MAC Generation



The OCE shall perform the same calculation on the response and use the same R-MAC session key employed by the Security Domain in order to verify the R-MAC.

6.5 APDU Command C-MAC and C-DECRYPTION Generation and Verification

Depending on the security level defined in the initiation of the Secure Channel, all subsequent APDU commands within the Secure Channel may require secure messaging and such as use of a C-MAC (integrity) and encryption (confidentiality). This section applies when both command confidentiality (C-DECRYPTION) and integrity (C-MAC) are required and the protocol in use does not implement an AEAD cipher (i.e. it will implement separate MAC and decryption functions).

For each APDU command sent within the secure channel session, the OCE shall increment an encryption counter:

- The encryption counter’s start value shall be set to 1 for the first command following a successful EXTERNAL AUTHENTICATE command.
- The encryption counter’s binary value shall be left padded with zeroes to form a full block of the block size of the cipher algorithm.
- This block shall be encrypted with S-ENC to produce the ICV for command encryption.

Note: This scheme fulfils the requirements described in [NIST 800-38A] for unpredictable ICVs when using CBC mode.

No encryption shall be applied to a command where there is no command data field. In this case, the encryption counter shall still be incremented as described above, and the message shall be protected as defined in section 6.3. Otherwise, the OCE performs the process detailed hereafter.

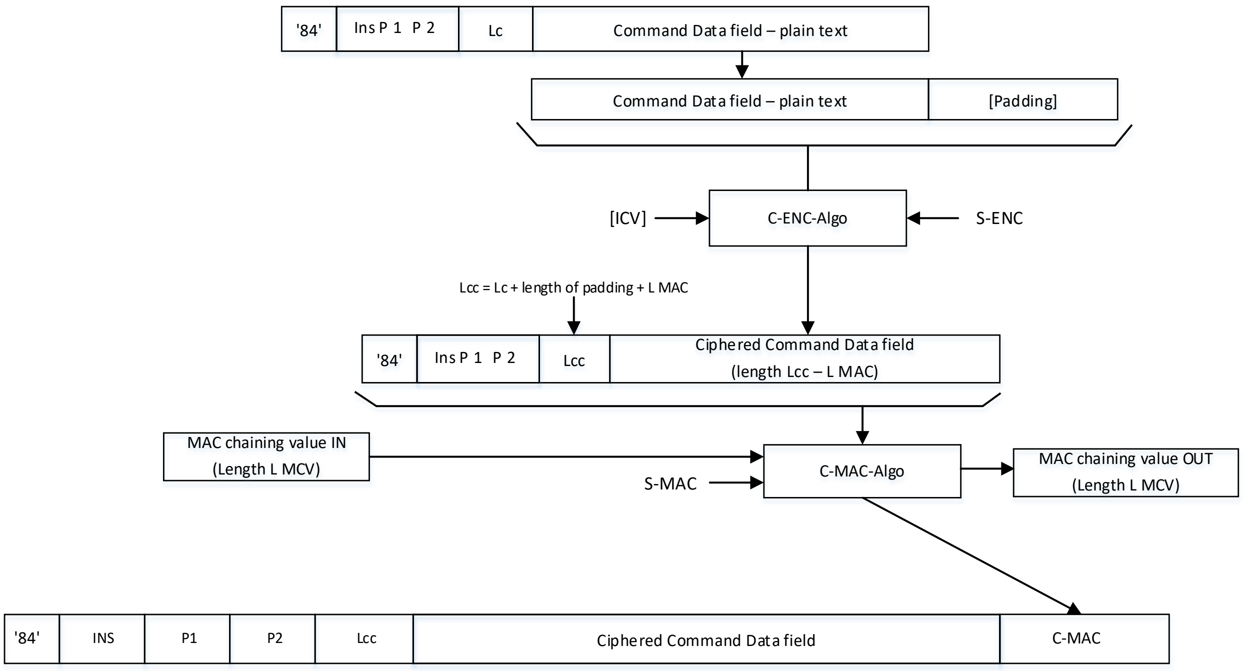
The OCE first encrypts the command data field using the cipher algorithm and then computes the C-MAC on the command with the ciphered data field as described in section 6.3.

Details of message encryption and decryption are given in the appropriate Protocol Function descriptions (see section 3.2.4). Padding is added to the input data as required by the cipher algorithm.

The final Lc value (Lcc) is the sum of:

initial Lc + length of the padding (as required) + length of C-MAC

Figure 6-3: APDU Command Data Field Encryption


'84'
Ins P 1 P 2
Lc
Command Data field – plain text
Command Data field – plain text
[Padding]
[ICV]
C-ENC-Algo
S-ENC
Lcc = Lc + length of padding + L MAC
'84'
Ins P 1 P 2
Lcc
Ciphered Command Data field (length Lcc – L MAC)
MAC chaining value IN (Length L MCV)
S-MAC
C-MAC-Algo
MAC chaining value OUT (Length L MCV)
'84'
INS
P1
P2
Lcc
Ciphered Command Data field
C-MAC

6.6 APDU Response R-MAC and R-ENCRYPTION Generation and Verification

This section applies when both response confidentiality (R-ENCRYPTION) and integrity (R-MAC) are required.

Depending on the security level defined in the initiation of the Secure Channel, all subsequent APDU responses within the Secure Channel may require secure messaging and such as use of an R-MAC (integrity) and encryption (confidentiality). No encryption shall be applied to a response where there is no response data field: In this case the message shall be protected as defined in section 6.4.

The Security Domain first encrypts the Response Data field and then computes the R-MAC on the response with the ciphered data field as described in section 6.4.

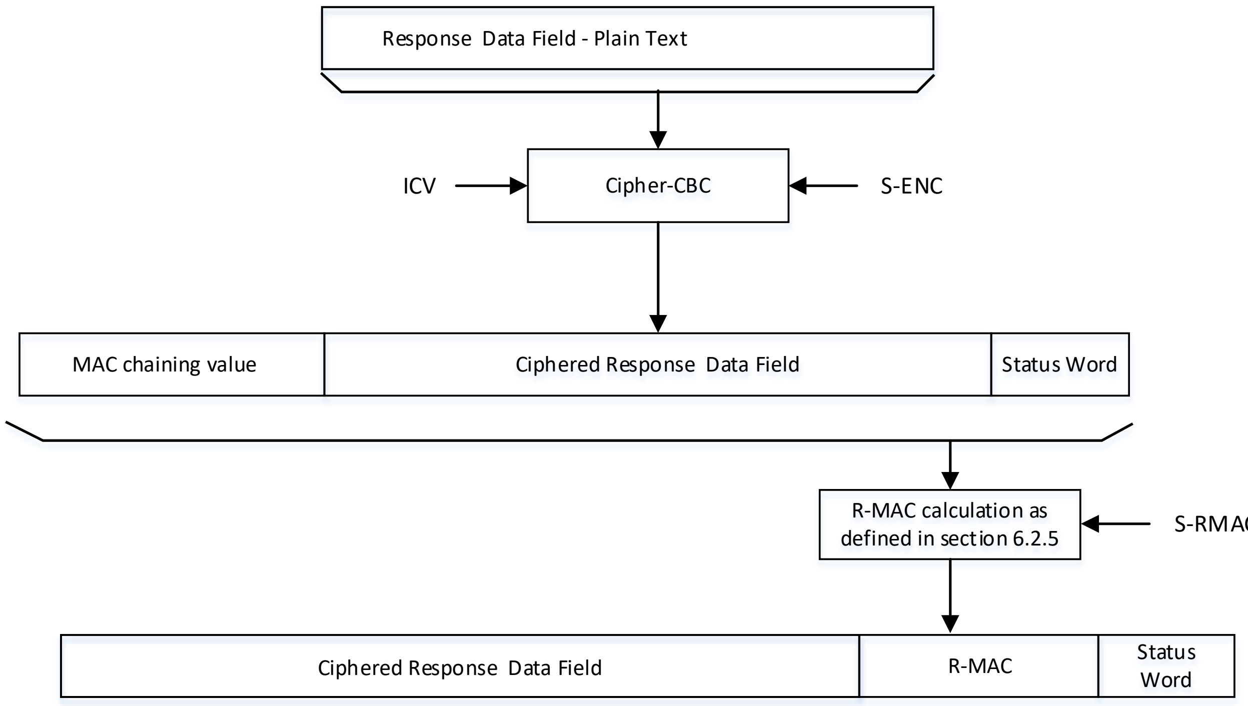
The response message encryption uses the protocol's cipher algorithm (see sections 3.1.4 and 3.2.4) with the following inputs:

- The key is the S-ENC session key.
- The ICV for the cipher algorithm shall be calculated as follows:
    - The padded counter block used for the generation of the ICV for command encryption shall also be used to generate the ICV for response encryption, however, with the following additional intermediate step: Before encryption, the most significant byte of this block shall be set to '80'.
    - This block shall be encrypted with S-ENC to produce the ICV for response encryption. The modification in the most significant byte guarantees that the ICVs for R-ENCRYPTION are different from those used for C-DECRYPTION.

  Note: This scheme fulfils the requirements described in [NIST 800-38A] for unpredictable ICVs when using CBC mode.
- The input data are the response data (padding might be applied as required by the cipher algorithm).

The final response APDU shall be the concatenation of the ciphered data, the R-MAC and the Status Word.

Figure 6-4: APDU Response Data Field Encryption


Response Data Field - Plain Text
ICV
Cipher-CBC
S-ENC
MAC chaining value
Ciphered Response Data Field
Status Word
R-MAC calculation as defined in section 6.2.5
S-RMAC
Ciphered Response Data Field
R-MAC
Status Word

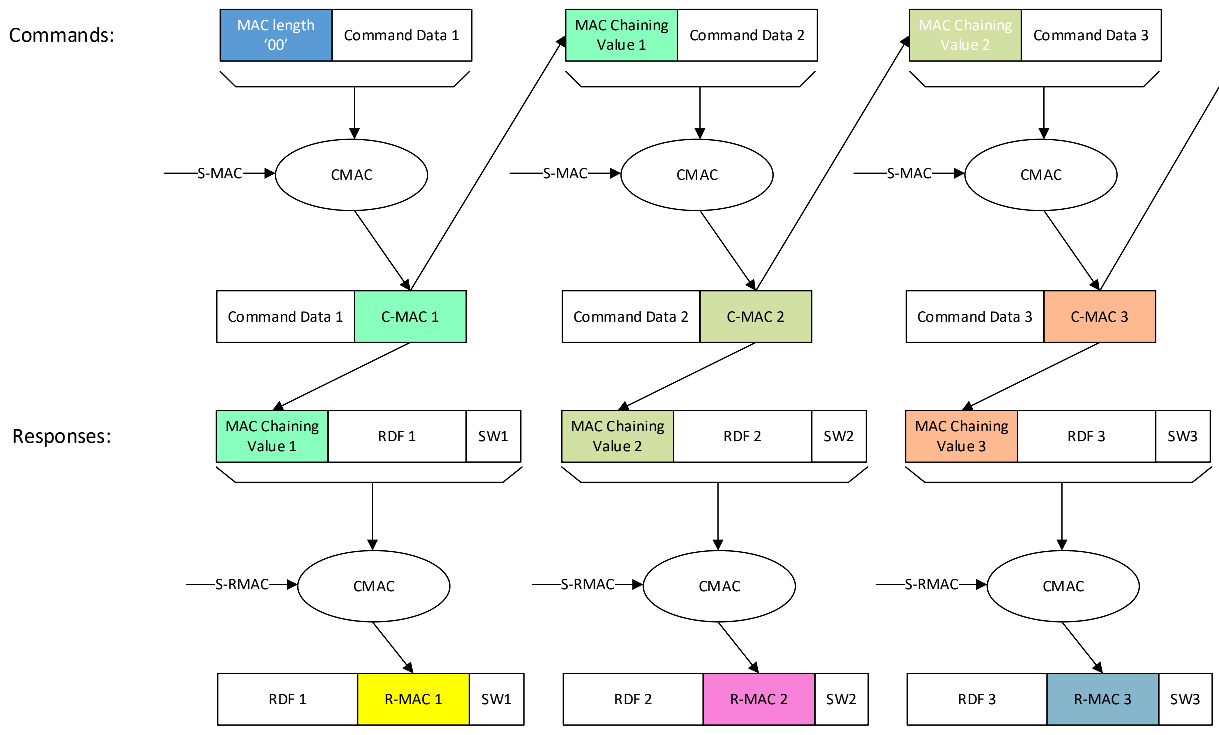
6.7 MAC Chaining

Consecutive commands are chained to each other by the incorporation of the MAC Chaining Value, the MAC associated with the previous command, protecting their sequence. The initial value for the first C-MAC to be generated (associated with the EXTERNAL AUTHENTICATE command) uses bytes of zeros (length given by the length of the MAC used by the protocol).

Responses are linked to the respective command using the MAC Chaining Value. As R-MAC uses a different session key from C-MAC, the same MAC Chaining Value can be used for the response and the next command.

Figure 6-5 illustrates the combined MAC chaining for command and responses.

Figure 6-5: MAC Chaining



Note that ~~if the~~ security levels ~~indicates~~ indicating R-MAC without C-MAC are not supported (see section 4.7)~~.then there will be no C-MAC to be used as the MAC Chaining Value. In this case:~~

~~The initial R-MAC Chaining Value is set to zeros (again, length given by the length of the MAC used by the protocol).~~

~~Subsequent R-MAC generation incorporates the previous R-MAC.~~

~~This mechanism then corresponds directly to the generation of C-MAC.~~

6.8 AEAD Initialization Vector

The security of AEAD algorithms is dependent on the uniqueness of the Initialization Vector (IV). Following the recommendation provided in [NIST 800-38D] for the RBG-based construction of the IV, it is to be generated for each invocation of the AEAD function as follows:

- The IV is composed of a random field, of at least 128 bits, and an empty free field.
- For the first invocation of the AEAD function (i.e. the verification of the C-MAC associated with the EXTERNAL AUTHENTICATE command), the IV shall be set to the Card Challenge associated with the INITIALIZE UPDATE command.
- For each subsequent invocation of the AEAD function, the IV shall be set to the value of the authentication tag provided by the previous use of the AEAD function (i.e. either the C-MAC or R-MAC from the previous command/response, depending on the security level).

Note that this mechanism for IV generation provides the same chaining function as MAC chaining does in the conventional use of CBC algorithms (for example using either AES or SM4). AEAD algorithms do not use the MAC Chaining mechanism described in section 6.7.

6.9 AEAD Combined Confidentiality and Authentication

6.9.1 Command Encryption and Authentication

In order to facilitate the use of AEAD with ISO 7816-4 commands, the modified APDU header is provided as the Associated Data to the AEAD cipher algorithm:

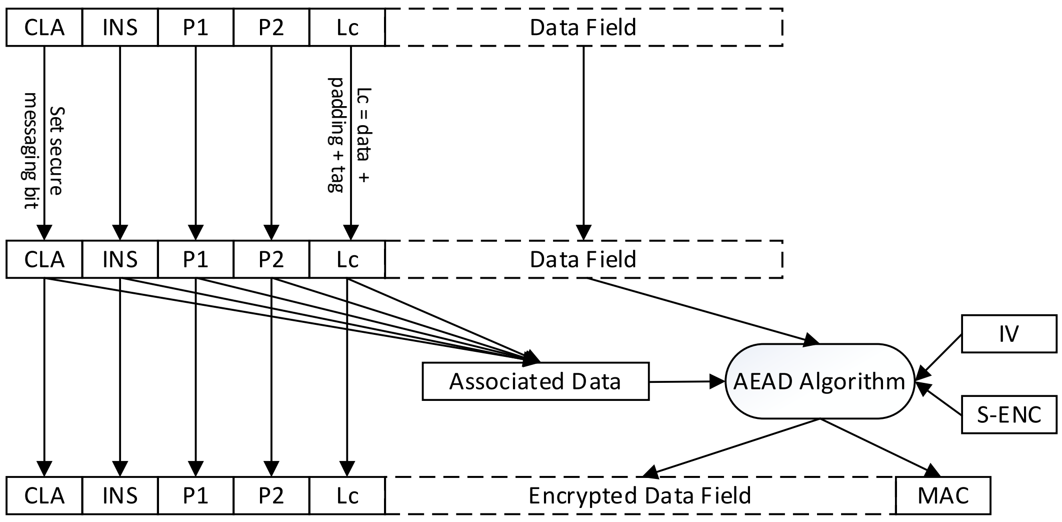
- The secure messaging bit of the CLA byte is set.
- The Lc parameter is adjusted to take into account any padding, if required by the encryption mode, ~~(the length of plaintext plus padding must be a multiple of the block length of the encryption algorithm)~~ and the length of the authentication tag.

The inputs to the cipher algorithm are the following:

- The IV (see section 6.8)
- The encryption key (S-ENC session key)
- The mode (ENCRYPT)
- The data field (to be ciphered)
- The associated data (APDU header)

The cipher algorithm is used to generate the encrypted data field and the authentication tag. The protected command is then constructed from the concatenation of the associated data (APDU header), the encrypted data field and the authentication tag (i.e. MAC). The process is shown in Figure 6-6.

Figure 6-6: AEAD Encryption and Authentication on Modified APDU



6.9.2 Combined Command Decryption and Authentication

The decryption and tag authentication operation is the reverse operation to encryption and authentication tag generation. The inputs are the following:

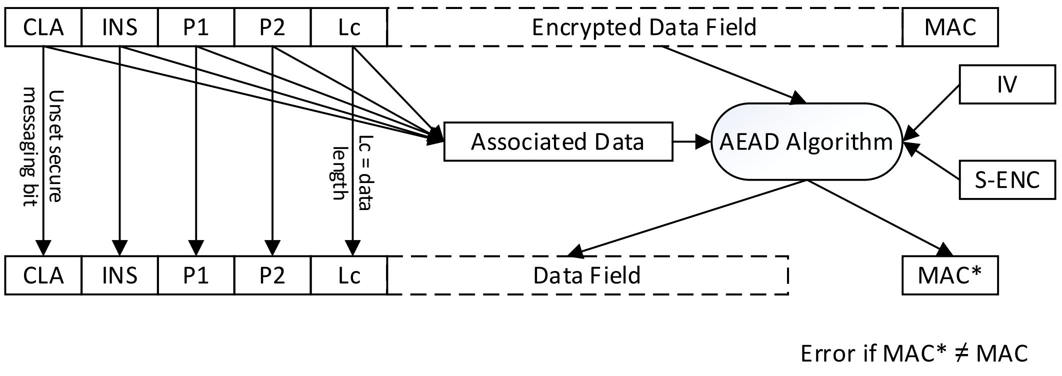
- The IV (see section 6.8)
- The encryption key (S-ENC session key)
- The mode (DECRYPT)
- The encrypted data field (ciphertext) and authentication tag (the length of the tag is implicitly known by the algorithm)
- The associated data (APDU header)

The process uses the cipher algorithm to decrypt the ciphertext and generate the corresponding authentication tag. It then compares the two tags – if they differ then authentication verification has failed.

If authentication is successful, the cipher algorithm will have decrypted the data and removed any padding that had been added to make the plaintext a multiple of encryption block lengths, leaving the plaintext in situ. The value of Lc is adjusted accordingly taking into account the removal of the authentication tag and any padding. The secure messaging bit is removed from the CLA byte.

Figure 6-7 illustrates the process:

Figure 6-7: AEAD Decryption and Authentication of Modified APDU



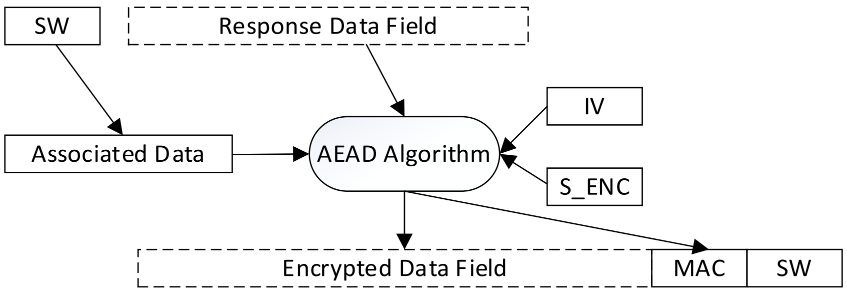
6.9.3 Response Encryption and Authentication

For AEAD algorithms the combined encryption and authentication of response data is performed by using the cipher algorithm with the following parameters:

- The IV is derived from the previous IV, as described in section 6.8.
- The encryption key (S-ENC session key)
- The mode (ENCRYPT)
- The response data field (if any, to be ciphered)
- The associated data is the status bytes.

Secure messaging is not applied to response data if the status bytes indicate an error. All status words, except '9000' and warning status words (i.e. '62xx' and '63xx'), shall be interpreted as error status words.

Figure 6-8: AEAD Response Encryption and Authentication

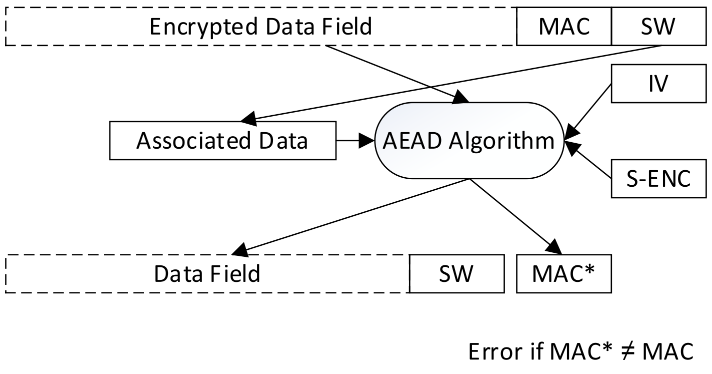


6.9.4 Response Decryption and Authentication

The AEAD response decryption and authentication process is shown in Figure 6-9.

Note that the only associated data are the status bytes.

Figure 6-9: AEAD Response Decryption and Authentication



6.9.5 AEAD MAC Generation and Verification

AEAD algorithms support MAC generation without data encryption by a process that is identical to the combined mode except that all of the data are considered to be associated data. The inputs to the cipher algorithm are the following:

- The IV (see section 6.8)
- The encryption key (S-ENC session key)
- The mode (MAC)
- No input data (i.e. no data will be ciphered)
- The associated data, comprising all of the data (i.e. APDU header and data field)

For APDU commands, the associated data is composed of the APDU header and the data field (if any) where the CLA and Lc bytes are modified prior to MAC generation, and are reset on MAC verification.

Figure 6-10: AEAD MAC Generation on Modified APDU

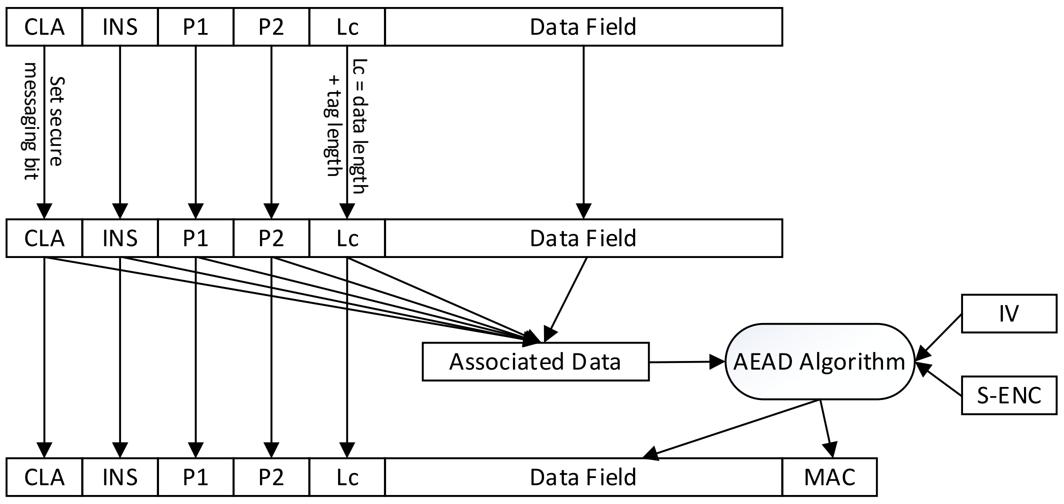
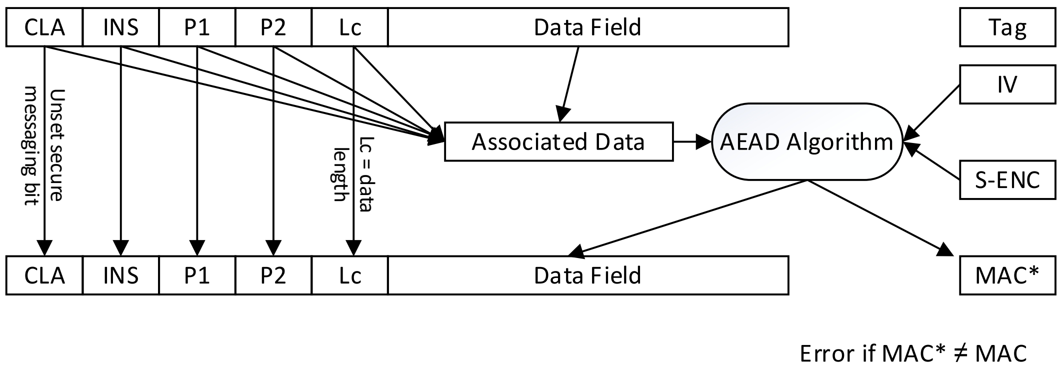


Figure 6-11: AEAD MAC Verification on Modified APDU



For APDU responses, the associated data is composed of the response data field (if any) and status bytes. The final response is formed from the concatenation of the following:

- The response data
- The authentication tag
- The status bytes

Secure messaging is not applied to response data if the status bytes indicate an error. All status words, except '9000' and warning status words (i.e. '62xx' and '63xx'), shall be interpreted as error status words.

Figure 6-12: AEAD Response MAC Generation

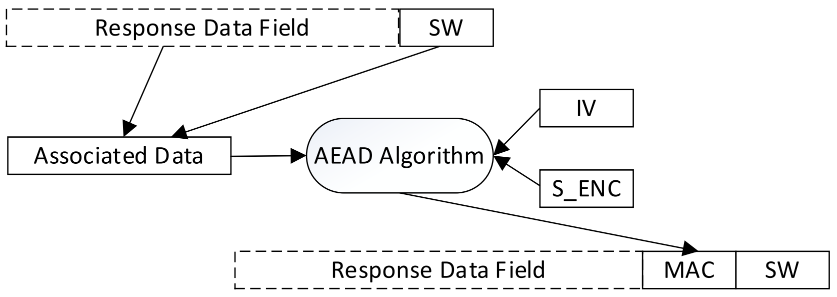
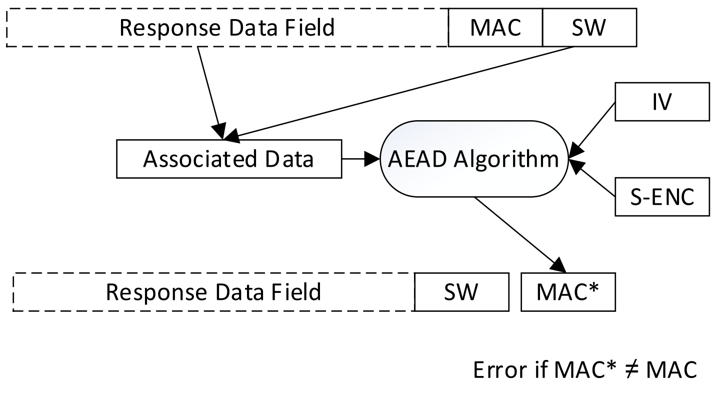


Figure 6-13: AEAD Response MAC Verification



6.10 Sensitive Data Encryption and Decryption

Key data encryption is used when transmitting sensitive data to the Security Domain and is over and beyond the security level required for the Secure Channel. For instance, all symmetric or private keys transmitted to a Security Domain should be encrypted.

The data encryption process uses the data encryption key (Key-DEK) and the Sensitive Data Decryption Protocol Function (see section 3.1.5).

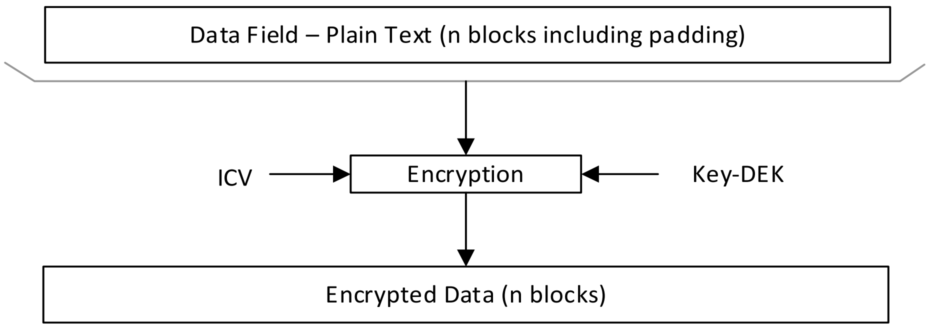
If the size of the sensitive data to be encrypted is a multiple of the block size (e.g. 16 bytes for AES), then no padding is required for the data field prior to encryption; otherwise the sensitive data are padded according to the specification of the protocol.

Note that if sensitive data encryption uses a different algorithm from that used for message integrity and confidentiality then the Key-DEK may differ in type from Key-ENC/MAC.

AES and SM4 encryption of sensitive data uses CBC encryption with ICV set to zero, performed across the sensitive data. The on-card decryption of key data is the exact opposite of the above operation.

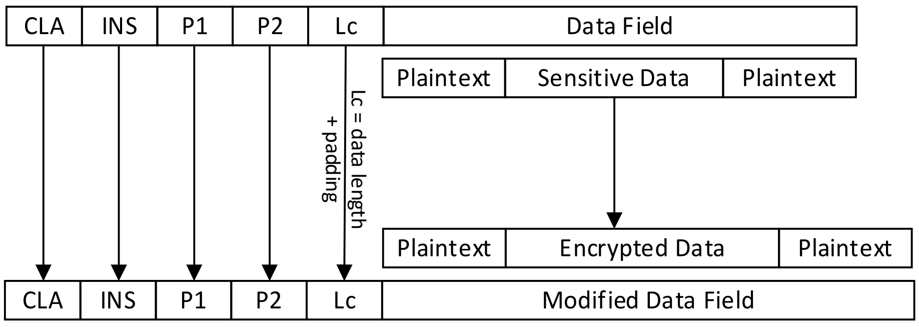
The encryption process is shown in Figure 6-14.

Figure 6-14: Sensitive Data Encryption


Data Field – Plain Text (n blocks including padding)
ICV
Encryption
Key-DEK
Encrypted Data (n blocks)

The encrypted data block then becomes part of the command data field, taking the place of the sensitive data, as shown in Figure 6-15.

Figure 6-15: Embedding Encrypted Sensitive Data in the APDU



The command is then processed according to the requirements of the current Secure Channel's security level (i.e. adding a MAC, or command data encryption and MAC generation).

7 COMMANDS

The following table presents the commands involved in Secure Channel Initiation.

Table 7-1: SCP04 Command Support

Command	Secure Channel Initiation
GET DATA	✓
INITIALIZE UPDATE	✓
EXTERNAL AUTHENTICATE	✓

Ticks (✓) denote that support of the command is mandatory.

Blank cells denote that the support of the command is optional.

7.1 GET DATA

The GET DATA command is defined in [GPCS] section 11.3. Bit b8 of the class byte shall be set to 1 – support for bit b8 set to 0 is optional. The instruction code shall be set to 'CA'.

To retrieve the Protocol Configuration List (see section 3.3.3), parameters P1 and P2 shall be set to '9~~B~~F71'.

> **Note:** The off-card entity uses the Protocol Configuration List to select a protocol among those supported by the Security Domain. This list is also used as an input to the session key derivation mechanism (see section 3.2.1) to ensure that the off-card entity has received the correct Protocol Configuration List.

For SCP04 keys, the Key Information Template (tag 'E0') defined in [GPCS] section 11.3.3.1.1 allows to retrieve the SCP identifier and Protocol Configuration Identifier(s) associated with the key. For this purpose, the value-part of TLV '97' (as described in Table 7-11) is added to the Extended Key Information Data Structure (see Table 7-2). For SCP04 keys, this Extended Key Information Data Structure shall be used.

Table 7-2: Key Information Data Structure – Extended

Name	**Length**	**Value**	**Presence**
Key Identifier	1	Key Identifier value	Mandatory
Key Version Number	1	Key Version Number value	Mandatory
Key type of first (or only) key component	2	'FF xx'	Mandatory
Length of first (or only) key component	2	'00 01' - '7F FF'	Mandatory
…	…	…	
Key type of last key component (if any)	2	'FF xx'	Conditional
Length of last key component (if any)	2	'00 01' - '7F FF'	Conditional
Length of Key Usage	1	'00' - '01'	Mandatory
Key Usage	0 or 1	'xx'	Conditional
Length of Key Access	1	'00' - '01'	Mandatory
Key Access	0 or 1	'xx'	Conditional
Length of SCP identifier and Protocol Configuration Identifier(s)	1	'00' - 'xx'	Mandatory
SCP identifier and Protocol Configuration Identifier(s)	1-n	SCP Identifier (i.e. '04') followed by a list of Protocol Configuration Identifiers associated with the key	Conditional

If the response exceeds 256 bytes, it shall be chained as described in [GPCS] section 11.1.5.2, using the '61xx' status word, and multiple GET RESPONSE commands should be used to retrieve the remaining response data.

7.2 INITIALIZE UPDATE

The INITIALIZE UPDATE command is used to start the process of mutual authentication, the first stage of setting up a secure channel.

The INITIALIZE UPDATE command message shall be coded according to the following table:

Table 7-3: INITIALIZE UPDATE Command Message

Code	Value	Meaning
CLA	'80' – '83', 'C0' – 'CF'	See [GPCS] section 11.1.4
INS	'50'	INITIALIZE UPDATE
P1	'xx'	KVN of the secure channel keyset
P2	'FF'	
Lc	Var.	Length of command data sent to Security Domain
Data	'xx xx…'	Protocol Configuration Identifier and Host Challenge sent to the Security Domain
Le	'00'	~~Length of expected response from Security Domain~~

7.2.1 Reference Control Parameter P1 – Key Version Number

The Key Version Number defines the Key Version Number within the Security Domain to be used to initiate the Secure Channel Session. If this value is zero, the first available key (starting with the default key) identified by the Security Domain as applicable for the ~~indicated protocol~~ Protocol Configuration Identifier (provided in the command data field) will be used. If a Protocol Configuration Identifier was provided in tag '97' when the key was loaded (see section 7.4), one of the Protocol Configuration Identifiers provided in tag '97' must match the Protocol Configuration Identifier of tag '80' in the data field. If no Protocol Configuration Identifier was provided when the key was loaded, the Key Type must be applicable for the Protocol Configuration Identifier of tag '80' in the data field; e.g. an AES key (Key Type of '88') is applicable for Protocol Configuration '01' and '03'.

Applicable Key Version Numbers for SCP04 may be defined in configuration documents.

If no applicable SCP04 Key Set is present within the Security Domain (e.g. no key set with matching Protocol Configuration Identifier), or if any of the mandatory keys listed in section 5.1 is missing in the targeted key set, then the INITIALIZE UPDATE command shall fail with error condition "Referenced data not found", as defined in section 7.2.5.

If the key access conditions don't allow access to any key of the identified SCP04 key set, the command is rejected with "Conditions of use not satisfied" as defined in [GPCS].

7.2.2 Reference Control Parameter P2 ~~– Protocol Identifier~~

Reference Control Parameter P2 is set to 'FF' to differentiate this command from the INITIALIZE UPDATE command of SCP03 (and the deprecated SCP02).

7.2.3 Data Field Sent in the Command Message

The data field of the INITIALIZE UPDATE command is described in the following table.

Table 7-4: Authentication Data Objects INITIALIZE UPDATE Command

Tag	Length	Value	Presence
'80'	~~Var.~~1 or 2	Protocol Configuration Identifier chosen by the OCE for the secure channel	Mandatory
'8A'	Var.	Host Challenge	Mandatory

If the data field is not formatted correctly, the command shall fail with error condition “Incorrect values in the command data” as defined in [GPCS].

If the value provided in tag '80' is not present in the Protocol Configuration List of the SD, the command shall fail with error condition “Incorrect values in the command data” as defined in [GPCS].

7.2.4 ~~Initialize Update~~ Response Data

The data field of the response message shall contain the following data elements in TLV format:

Table 7-5: INITIALIZE UPDATE Response Message

Tag	Length	Value			Presence
'A0'	Var.				Mandatory
		Tag	**Length**	**Value**	
		'90'	Var.	Key Diversification Data	Conditional (see below)
		'91'	4 or 5 bytes	Key Information	Mandatory
		'92'	3 bytes	Sequence Counter	Conditional (~~pseudo-random~~ predictable card challenge)
		'8B'	Var.	Card Challenge	Mandatory
		'8C'	Var.	Card Cryptogram	Mandatory
		'9F71'	Var.	Protocol Configuration List	Mandatory

The presence of Key Diversification Data is dependent on the data having been previously stored in the Security Domain (for instance via STORE DATA with DGI '00CF'). The data are typically used by a backend system (OCE) to derive the Security Domain's ~~Static Keys~~ static keys. The content of the Key Diversification Data is out of scope of this specification; the Security Domain treats the data as a blob of information that it does not process itself, but it may be retrieved using GET DATA (with the tag 'CF').

The Key Information is composed of the following single byte items:

- The Secure Channel Protocol Identifier, here '04'
- The Protocol Configuration Identifier (which must have the same value as the Protocol Configuration Identifier provided in the value-part of TLV '80' in the command data ~~the P2 parameter~~)
- The Secure Channel Protocol "i" parameter
- The Key Version Number used in initiating the Secure Channel Session

The Sequence Counter is only present when SCP04 is configured for ~~pseudo-random~~ predictable card challenge generation.

The Security Domain attaches the Card Challenge and Card Cryptograms, generated as specified in section 6.

7.2.5 Processing State Returned in the Response Message

A successful execution of the command shall be indicated by status bytes '90 00'.

This command may return either a general error condition as listed in [GPCS] section 11.1.3, General Error Conditions, or the following error condition:

Table 7-6: INITIALIZE UPDATE Error Conditions

SW1	SW2	Meaning
'6A'	'88'	Referenced data not found
'6A'	'80'	Incorrect values in the command data

7.3 EXTERNAL AUTHENTICATE

7.3.1 Definition and Scope

The EXTERNAL AUTHENTICATE command is used by the Security Domain, during explicit initiation of a Secure Channel, to authenticate the OCE and to determine the level of security required for all subsequent commands.

A successful execution of the INITIALIZE UPDATE command shall precede this command.

7.3.2 Command Message

The EXTERNAL AUTHENTICATE command message shall be coded according to the following table:

Table 7-7: EXTERNAL AUTHENTICATE Command Message

Code	Value	Meaning
CLA	'84' – '87', 'E0' – 'EF'	See [GPCS] section 11.1.4
INS	'82'	EXTERNAL AUTHENTICATE
P1	'xx'	Security Level
P2	'00'	
Lc	Var.	Length of authentication data sent to the Security Domain
Data	'xx xx…'	Authentication data sent to the Security Domain
Le	~~'00'~~	~~Length of expected response from the Security Domain.~~Not present

7.3.3 Reference Control Parameter P1 – Security Level

The reference control parameter P1 defines the level of security for all secure messaging commands within this Secure Channel Session following this EXTERNAL AUTHENTICATE command (it does not apply to this command).

Table 7-8: Security Level Encoding

b8	b7	b6	b5	b4	b3	b2	b1	Description
0	0	1	1	0	0	1	1	C-DECRYPTION, C-MAC, R-ENCRYPTION, and R-MAC
0	0	0	1	0	0	1	1	C-DECRYPTION, C-MAC, and R-MAC
0	0	0	1	0	0	0	1	C-MAC and R-MAC
0	0	0	0	0	0	1	1	C-DECRYPTION and C-MAC
0	0	0	0	0	0	0	1	C-MAC
0	0	0	0	0	0	0	0	No secure messaging expected
all other values								RFU

If the requested security level is not supported by the SD (see section 4.1), then the command is rejected with “Incorrect P1 P2” as defined in [GPCS].

7.3.4 Data Field Sent in the Command Message

The data field of the command message contains the Host Cryptogram and the C-MAC.

Table 7-9: EXTERNAL AUTHENTICATE Command Data Field

Tag	Length	Value	Presence
'8D'	Var.	Host Cryptogram	Mandatory
-	-	C-MAC	Mandatory

The C-MAC shall be present, irrespective of the security level specified by the P1 parameter, and shall be generated using the MAC Protocol Function of the selected Protocol Configuration and the appropriate session key.

7.3.5 Processing State Returned in the Response Message

A successful execution of the command shall be indicated by status bytes '90 00'.

This command may return either a general error condition as listed in [GPCS] section 11.1.3, General Error Conditions, or the following error condition:

Table 7-10: EXTERNAL AUTHENTICATE Error Conditions

SW1	SW2	Meaning
'63'	'00'	Authentication of host cryptogram failed.

7.4 STORE DATA Command (SCP04 Keys)

The STORE DATA command (see [GPCS] section 11.11) shall be used with DGI '00B9' to ~~install~~ load or replace SCP04 ~~Static Keys~~ keys~~, and DAP, Token, and Receipt keys (see section 7.5)~~. The Key Control Reference Template (DGI '00B9') is optionally extended for SCP04 keys to include the Protocol Configuration Identifier(s) with which they are associated:

Table 7-11: Data Content for DGI '00B9'

Tag	Length	Description	Presence
'B9'	Var.	CRT tag (CT)	Mandatory
'95'	1	Key Usage Qualifier according to [GPCS] section 11.1.9	Mandatory
'96'	1	Key Access according to [GPCS] Table 11-18	Optional
'80'	1	Key Type according to [GPCS] Table 11-16	Mandatory
'81'	1 or 2	Key Length, in bytes (unsigned integer value)	Mandatory
'82'	1	Key Identifier	Mandatory
'83'	1	Key Version Number	Mandatory
'84'	3	Key check value	Mandatory
'97'	1-n	SCP identifier and Protocol Configuration Identifier(s)	Optional

Tag '97', if present, shall contain the SCP identifier (i.e. '04') and may optionally contain a list of one or more Protocol Configuration Identifiers. When one or more Protocol Configuration Identifiers are present in tag '97', use of SCP04 keys is limited to the listed Protocol Configuration(s). The recipient Security Domain shall check that:

- each listed Protocol Configuration Identifier is supported by the Security Domain (appears in its Protocol Configuration List; see section 7.5), and
- the key type is appropriate for the listed Protocol Configuration(s).

The key check value for keys shall be calculated as specified in [GPCS] section B-6 for AES and SM-4 keys. For SM-4 the constant '02' shall be used in the generation of the key check value.

The DGI '8113' shall immediately follow DGI '00B9' and is used to populate the SCP04 key.

Table 7-12: Data Content for DGI '8113'

DGI	Length	Data Content	Encrypt
'8113'	Var.	SCP04 ~~Static Key~~ key	Yes

Note: It is not possible to update the attributes (Key Usage Qualifier, Key Access conditions, Key Type, Key Length, KVN/KID, SCP identifier and Protocol Configuration Identifier(s)) of an existing SCP04 key set. In case changing key attributes is required, a new key set must be stored. The old key set may be deleted with a DELETE [key] command.

7.5 STORE DATA (Protocol Configuration List)

This command shall be coded according to [GPCS] section 11.11~~.4~~, with the following additional requirements:

- The DGI '9~~B~~F71' shall be used to ~~provide~~ store or replace the Protocol Configuration List, as defined in section 3.3.3. E.g. '9F71', indicating support for Protocol Configuration '01' and '02', is encoded as '9F71020102'. Protocol Configuration Identifiers may occur in arbitrary order in the Protocol Configuration List.

The recipient Security Domain is responsible for ensuring that its Protocol Configuration List only contains protocols supported by the implementation ~~SE~~.

If any Protocol Configuration Identifier in the Protocol Configuration List is not supported by the implementation, or if a Protocol Configuration Identifier occurs more than once, the command shall fail with error condition “Incorrect values in command data” as defined in [GPCS] section 11.11.

The Security Domain’s Protocol Configuration List may be retrieved using the GET DATA command; see section 7.1.

7.6 PUT KEY

The PUT KEY command should ~~is~~ not be used to load nor replace SCP04 keys. The handling of a PUT KEY command that attempts to load or replace an SCP04 key is implementation specific.