Global Platform®

The standard for secure digital services and devices

GlobalPlatform Technology

Secure Channel Protocol '11'

Card Specification v2.3 – Amendment F

Version 1.3.0.13

Public Review

August 2023

Document Reference: GPC_SPE_093

THIS SPECIFICATION OR OTHER WORK PRODUCT IS BEING OFFERED WITHOUT ANY WARRANTY WHATSOEVER, AND IN PARTICULAR, ANY WARRANTY OF NON-INFRINGEMENT IS EXPRESSLY DISCLAIMED. ANY IMPLEMENTATION OF THIS SPECIFICATION OR OTHER WORK PRODUCT SHALL BE MADE ENTIRELY AT THE IMPLEMENTER'S OWN RISK, AND NEITHER THE COMPANY, NOR ANY OF ITS MEMBERS OR SUBMITTERS, SHALL HAVE ANY LIABILITY WHATSOEVER TO ANY IMPLEMENTER OR THIRD PARTY FOR ANY DAMAGES OF ANY NATURE WHATSOEVER DIRECTLY OR INDIRECTLY ARISING FROM THE IMPLEMENTATION OF THIS SPECIFICATION OR OTHER WORK PRODUCT.

Contents

Tables

Figures

1 INTRODUCTION

This document specifies a new secure channel protocol, named Secure Channel Protocol '11' (SCP11), based on Elliptic Curve Cryptography (ECC) for mutual authentication and secure channel initiation and on AES for secure messaging.

1.1 Audience

This amendment is intended primarily for card manufacturers and application developers developing GlobalPlatform card implementations.

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

This section lists references applicable to this specification. The latest version of each reference applies unless a publication date or version is explicitly stated.

Table 1-1: Normative References

Standard / Specification	Description	Ref
GPCS	GlobalPlatform Technology Card Specification v2.3.1 Document Reference: GPC_SPE_034	[GPCS]
GPCS Amendment A	GlobalPlatform Technology Confidential Card Content Management Card Specification v2.3 – Amendment A v1.2 Document Reference: GPC_SPE_007	[Amd A]
GPCS Amendment C	GlobalPlatform Technology Contactless Services Card Specification v2.3 – Amendment C v1.3 Document Reference: GPC_SPE_025	[Amd C]
GPCS Amendment D	GlobalPlatform Technology Secure Channel Protocol 03 Card Specification v2.3 – Amendment D v1.2 Document Reference: GPC_SPE_014	[Amd D]
BSI TR-03111, Version 2.0	BSI Technical Guideline TR-03111: Elliptic Curve Cryptography	[TR 03111]

Standard / Specification	Description	Ref
NIST SP 800-56A Revision 2	Recommendation for Pair-Wise Key-Establishment Schemes Using Discrete Logarithm Cryptography, May 2013	[NIST 800-56A]
RFC 4519	Lightweight Directory Access Protocol (LDAP)	[RFC 4519]
RFC 5280	Internet X.509 PKI Certificate and CRL Profile RFC 5280	[RFC 5280]
RFC 5480	RFC 5480 Elliptic Curve Cryptography Subject Public Key Information	[RFC 5480]
RFC 5639	Elliptic Curve Cryptography (ECC) Brainpool Standard Curves and Curve Generation	[RFC 5639]
RFC 5758	RFC 5758 Internet X.509 Public Key Infrastructure: Additional Algorithms and Identifiers for DSA and ECDSA	[RFC 5758]
ITU-T X.520	ITU-T X.520 Information technology – Open Systems Interconnection – The Directory: Selected attribute types	[ITU X520]

Table 1-2: Informative References

Standard / Specification	Description	Ref
Cryptographic Algorithm Recommendations	GlobalPlatform Technology Cryptographic Algorithm Recommendations Document Reference: GP_TEN_053	[Crypto Rec]
TEE Trusted User Interface Low-level API	GlobalPlatform Technology TEE Trusted User Interface Low-level API Document Reference: GPD_SPE 055	[TEE TUI Low]

1.4 Terminology and Definitions

Terms used in this document are generally defined in [GPCS].

Additional terms are listed in Table 1-3.

Table 1-3: Terminology and Definitions

Term	Definition
Card Group ID	A common identifier for a group of cards sharing the same CERT.SD.ECKA
Controlling Authority	In the context of this specification, an authority delivering certificates (root of a certification chain)
Key Authority	Authority delivering certificates (involved in a certification chain)

1.5 Abbreviations and Notations

Abbreviations and notations used in this document are included in Table 1-4.

Table 1-4: Abbreviations and Notations

Abbreviation / Notation	Meaning
AES	Advanced Encryption Standard
APDU	Application Protocol Data Unit
API	Application Programming Interface
BCD	Binary Coded Decimal
BER	Basic Encoding Rules
CA	Controlling Authority
CA-KLCC	Controlling Authority for Confidential Key Loading Card Certificates
CA-KLOC	Controlling Authority for Confidential Key Loading OCE Certificates
C-DECRYPTION	Command Decryption
C-ENC	Command Encryption
CERT.KA-KLCC.ECDSA	Certificate containing the public key of a KA-KLCC used for verifying certificates (CERT.SD.ECKA or another intermediate CERT.KA-KLCC.ECDSA)
CERT.KA-KLOC.ECDSA	Certificate containing the public key of a KA-KLOC used for verifying certificates (CERT.OCE.ECKA or another intermediate CERT.KA-KLOC.ECDSA)
CERT.OCE.ECKA	Certificate containing the public key of the OCE used for key agreement
CERT.SD.ECKA	Certificate containing the public key of the SD used for key agreement
CGM	Cumulative Granted Memory
CLA	CLAss byte of command message
C-MAC	Command MAC
CRT	Control Reference Template
CSN	Certificate Serial Number
DGI	Data Grouping Identifier
EC	Elliptic Curve
ECC	Elliptic Curve Cryptography
ECDSA	Elliptic Curve Digital Signature Algorithm
ECKA	Elliptic Curve Key Agreement
ELF	Executable Load File
ePK.OCE.ECKA	Ephemeral public key of the OCE used for key agreement
ePK.SD.ECKA	Ephemeral public key of the SD used for key agreement
eSK.OCE.ECKA	Ephemeral private key of the OCE used for key agreement

Abbreviation / Notation	Meaning
eSK.SD.ECKA	Ephemeral private key of the SD used for key agreement
INS	INStruction byte of command message
KA	Intermediate Key Authority used by CA to build certification chains
KA-KLCC	Intermediate Key Authority for Confidential Key Loading Card Certificates
KA-KLOC	Intermediate Key Authority for Confidential Key Loading OCE Certificates
Key-DEK	Data Encryption Key
KID	Key Identifier
KVN	Key Version Number
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
MAC	Message Authentication Code
MOC	Mandatory, Optional, Conditional
OCE	Off-Card Entity
P1	Reference control Parameter 1
P2	Reference control Parameter 2
PIN	Personal Identification Number
PK.CA-KLCC.ECDSA	Public key of the CA-KLCC used for verifying certificates
PK.CA-KLOC.ECDSA	Public key of the CA-KLOC used for verifying certificates
PK.KA-KLCC.ECDSA	Public key of the KA-KLCC used for verifying certificates
PK.KA-KLOC.ECDSA	Public key of the KA-KLOC used for verifying certificates
PK.OCE.ECKA	Public key of the OCE used for key agreement
PK.SD.ECKA	Public key of the SD used for key agreement
R-ENCRYPTION	Response Encryption
RFU	Reserved for Future Use
R-MAC	Response MAC
SCP	Secure Channel Protocol
SD	Security Domain
S-DEK	Session Data Encryption Key
S-ENC	Secure Channel session key for command and response encryption
ShS	Shared Secret (concatenated shared secrets, depending on SCP11 version a/b/c)
ShSee	Shared Secret calculated from two ephemeral keys
ShSes	Shared Secret calculated from one ephemeral (OCE) and one static (SD) key

Abbreviation / Notation	Meaning
ShSse	Shared Secret calculated from one static (OCE) and one ephemeral (SD) key
ShSss	Shared Secret calculated from two static keys
SK.CA-KLCC.ECDSA	Private key of the CA-KLCC used for signing certificates
SK.CA-KLOC.ECDSA	Private key of the CA-KLOC used for signing certificates
SK.KA-KLCC.ECDSA	Private key of the KA-KLCC used for signing certificates
SK.KA-KLOC.ECDSA	Private key of the KA-KLOC used for signing certificates
SK.OCE.ECKA	Private key of the OCE used for key agreement
SK.SD.ECKA	Private key of the SD used for key agreement
S-MAC	Secure Channel C-MAC session key
S-RMAC	Secure Channel R-MAC session key
TLV	Tag Length Value
UTF-8	Unicode Transformation Format – 8-bit
Var	Variable

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 1-5: Revision History

Date	Version	Description
May 2015	1.0	Public Release
Sep 2017	1.1	Public Release Added support for certificate chains down to CERT.OCE.ECKA and down to CERT.SD.ECKA. The root certificate authorities are still known as CA-KLOC and CA-KLCC respectively, and intermediate certificate authorities are known as KA-KLOC and KA-KLCC (i.e. Key Authorities). The GET DATA (ECKA Certificate) command can now return a chain of (on-card) certificates (i.e. Certificate Store). The PERFORM SECURITY OPERATION command can now present a chain of (off-card) certificates. See sections 3.3, 6.2, 6.4, and 6.7. Added support for storing several PK.CA-KLOC.ECDSA, each one corresponding to a specific CA-KLOC Identifier. See sections 5.1, 6.3, and 6.9.

Date	Version	Description
Jul 2018	1.2	Public Release Added SCP11c, which allows the offline creation of secured command sequences (scripts) for the SE. If SEs are personalized with group keys, the same script can be used on a group of SEs. SCP11c also specifies a transaction mechanism which allows a rollback in case of failures.
Mar 2019	1.2.1	Maintenance Release Clarified and fixed a number of errors in Annex B (Usage of SCP11c). Miscellaneous clarifications about SCP11c all across the document. Added warning notice in section 4.3 about using certificate allowlists as a revocation mechanism. Clarifications + additional rules in section 4.6 (Session Replay). More security levels available for SCP11c (section 4.7 and 6.5). Clarifications for certificate chaining in sections 6.2, 6.4, and 6.7. Other miscellaneous clarifications.
Oct 2021	1.3	Public Release Alignment with changes for SCP03: Support of 16-byte MAC. Clarification regarding the handling of the Subject ID. Clarifications regarding the usage of the API (section 4.7 (API and Security Level)). Clarifications regarding when OCE authorizations shall apply (tag 'BF20' of OCE certificate, section B.2 (Limitations on Commands Received through SCP11c).
Dec 2022	1.3.0.1	Member Review Add support for X.509 certificate format. Extend support of authorization rules (tag 'BF20') to SCP11a. SCP11 “i” parameter defines new options and can now be encoded on 1 or 2 bytes.

Date	Version	Description
Aug 2023	1.3.0.13	Public Review Many clarifications and errors fixed. Clarifications for the `SecureChannel.processSecurity(..)` method. New case described for STORE DATA command to store a CA-KLCC identifier together with the KID/KVN of associated SK.SD.ECKA. Previously, the only case described was to store a CA-KLOC identifier and KID/KVN of associated PK.CA-KLOC.ECDSA. New GET DATA command to retrieve all the CA-KLOC (resp. CA-KLCC) identifiers known to an SD (i.e. all at once), each with the KID/KVN of associated PK.CA-KLOC.ECDSA (resp. SK.SD.ECKA). The Card Content Transaction mechanism and SD Self-Deletion mechanism previously described specifically for SCP11c are now described in a more generic way and applicable to any kind of secure channel (potentially other than SCP11). Support is mandatory within SCP11c sessions (for backward compatibility) and optional within other kinds of secure channels. Cumulative Deletion (defined in [Amd C]) is here extended to allow cumulatively deleting an SD (and sub-hierarchy) at any level, not only at root level. The SD performing this operation must have the Global Delete privilege.
TBD	v1.4	Public Release

2 SECURE CHANNEL PROTOCOL '11'

2.1 Scope of the Document

This document specifies a secure channel protocol, named Secure Channel Protocol '11' (SCP11), based on Elliptic Curve Cryptography (ECC) for mutual authentication and secure channel initiation and on AES for secure messaging.

SCP11 reuses cryptographic mechanisms defined in GPCS Amendment A: Confidential Card Content Management ([Amd A]) and in GPCS Amendment D: SCP03 ([Amd D]).

Three variants of the protocol are defined:

- SCP11a provides mutual authentication between the Off-Card Entity (OCE) and the card.
- SCP11b provides authentication of the card to the OCE only. Authentication of the OCE to the card must be provided by other means; an example is provided in Annex A.
- SCP11c provides mutual authentication between the OCE and the card, using only static keys for the authentication of the card. This allows for offline scripting usage, authorized by the CERT.OCE.ECKA, as described in Annex B.

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

2.2 Use Cases and Requirements

Compared to SCP03, this protocol allows authentication and secure channel initiation based on certificates instead of pre-shared keys. This provides greater flexibility in cases where the two entities setting up the secure channel are not deployed in strict pairs.

ECC provides suitable security strength for the establishment of session keys for all three variants of AES: AES-128, AES-192, and AES-256.

SCP11c provides mutual authentication between the Off-Card Entity (OCE) and each card out of a group of cards.

- This variant can be used in scripting mode, i.e. it is possible to precompute personalization scripts for a group of cards which can be deployed later via online services or as resource data in companion apps to the entire group of cards. In this use case, the script is distributed as part of a rich environment application. Upon installation of such an application, the script is executed. No secret data is handled by the rich environment application and the cryptographic part of the OCE is run in a secure server of the OCE.
- The response (i.e. sequence of APDU responses) may be collected by the OCE, e.g. a rich environment application, and sent back to the service provider who can decrypt them using the SCP03 session keys, if required.
- If the Security Domain performs onboard key generation according to scenario #1 or scenario #3 from [Amd A], then the rich environment application stores the responses from the key generation commands and sends them back to the service provider. The service provider uses these responses to calculate the onboard generated key. This process works exactly as described in [Amd A].

For both SCP11a and SCP11c, the OCE certificate may contain Authorization Rules. Annex B provides details on what authorizations may be described and when they shall apply.

3 SPECIFICATION AMENDMENTS

3.1 Algorithms

This specification combines algorithms already specified in [Amd D] and [Amd A]. However, compared to [Amd A], SCP11 uses different input data for ECKA and the Key Derivation.

3.1.1 ECKA

An Elliptic Curve Key Agreement Algorithm (ECKA) is used in this specification for the establishment of session keys. A description of such schemes can be found e.g. in BSI Technical Guideline TR-03111 ([TR 03111]).

ECKA used in this specification shall follow the definition for the Key Agreement Algorithm in [TR 03111]. The algorithm is executed for each variant of the protocol, as follows:

- SCP11a: With two pairs of static keys and two pairs of ephemeral keys. This scheme is described in [NIST 800-56A] as “(Cofactor) Full Unified Model, C(2e, 2s, ECC CDH)” for curves with a cofactor of 1.
- SCP11b: With two ephemeral key pairs and one static key pair. The OCE does not have a static key pair. The ephemeral key pair of the OCE is used twice.
- SCP11c: With two static key pairs and one ephemeral key pair. The SD does not create an ephemeral key pair. Instead, the static key pair of the SD is used twice. This scheme is described in [NIST 800-56A] as “One-Pass Unified Model, C(1e, 2s, ECC CDH) Scheme”.

  Note: *The recommendations in [NIST 800-56A] on the handling of ephemeral keys and of intermediate results (e.g. the shared secrets ShSee, ShSes, ShSse, and ShSss) should be taken into account in an implementation.*

  Note: *Performing all the checks specified in [TR 03111] (including the check that the secret points are not zero) is required to avoid attacks on ephemeral public keys.*

3.1.2 Key Derivation

The shared secret ShS generated by Key Agreement Algorithm is not used directly as a key for cryptographic operations, but as an input to a key derivation process.

A key for calculating a receipt and the session keys are derived from the shared secret as defined in [TR 03111] for the “X9.63 Key Derivation Function”. This key derivation includes additional information, the “SharedInfo” of the key derivation algorithm.

3.2 Controlling Authority Roles

Within the context of SCP11, the Controlling Authority (CA) has two different roles:

- Providing certificates for the SD: CERT.SD.ECKA
- Providing certificates for the OCE: CERT.OCE.ECKA

As there is no technical need that one actor provides both roles, those roles are distinguished in this document:

- CA-KLCC denotes the role providing certificates for the SD on the Card: Controlling Authority for Confidential Key Loading Card Certificates.
- CA-KLOC denotes the role providing certificates for the OCE: Controlling Authority for Confidential Key Loading OCE Certificates.

Instead of directly using the Controlling Authority (CA), it may simplify deployment to have a subordinate Key Authority (KA) to which the CA may delegate the diversification of the keys and related certificates by setting up a certificate chain. As there is no technical need for a single actor to provide both the Off-Card Entity and Security Domain roles, those roles are distinguished in this document:

- KA-KLCC denotes the role providing certificates for the SD on the Card: Key Authority for Confidential Key Loading Card Certificates.
- KA-KLOC denotes the role providing certificates for the OCE: Key Authority for Confidential Key Loading OCE Certificates.

To simplify the description of the mechanisms, we will keep using the abbreviation CA in the rest of this document, even if one or more Key Authorities are involved and a certificate chain is used.

3.3 Certificate Verification

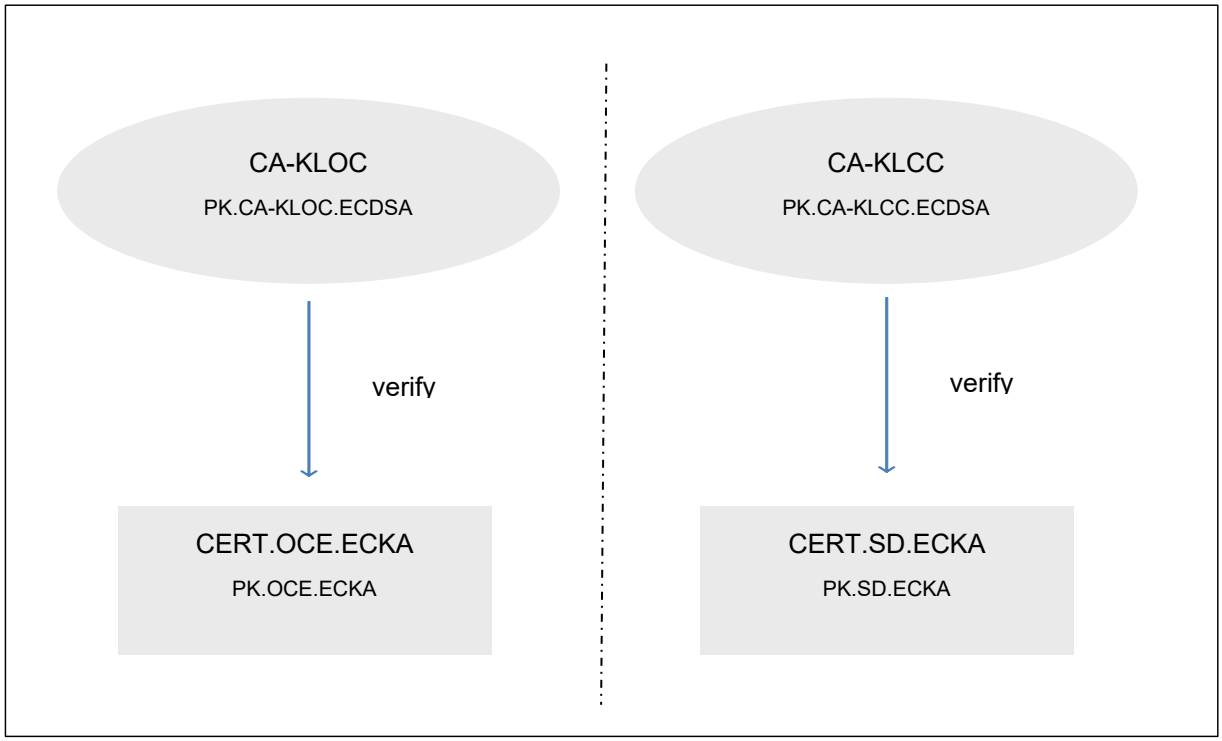
The Security Domain and the OCE may need to traverse and verify a chain of certificates from established trust points down to each other's public key.

By default, the Security Domain expects the first certificate presented for verification to be signed by the CA-KLOC, and each subsequent certificate to be signed by the public key validated with the previous certificate.

By default, the OCE expects the first certificate presented for verification to be signed by the CA-KLCC, and each subsequent certificate to be signed by the public key validated with the previous certificate.

The example shown in Figure 3-1 is a simple case where the CA-KLOC directly certifies the public key of the Off-Card Entity and the CA-KLCC directly certifies the public key of the Security Domain.

Figure 3-1: Single Certificate (No Chaining)

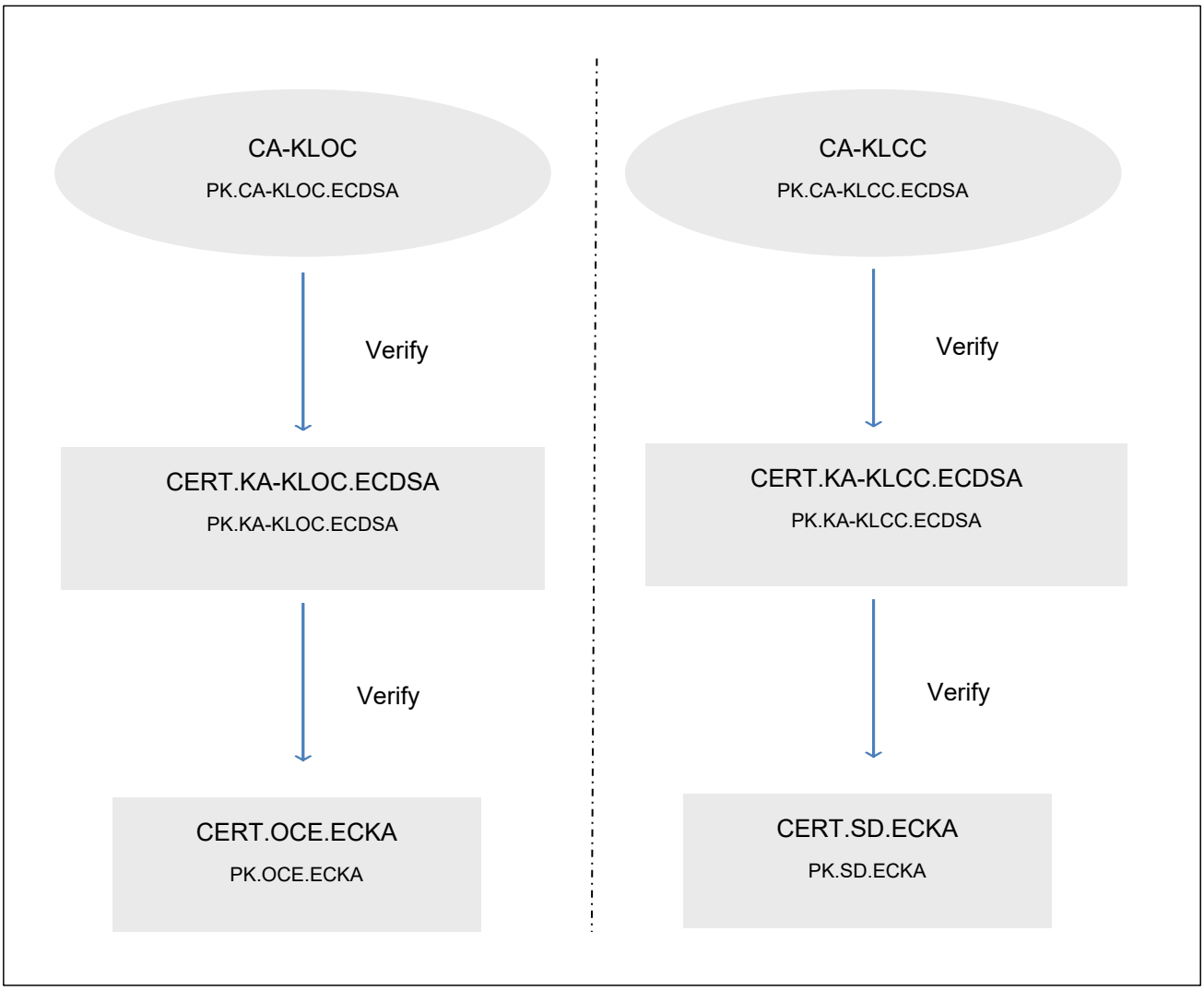


The example shown in Figure 3-2 illustrates the usage of a certificate chain.

The CA-KLOC certifies the KA certificate (CERT.KA-KLOC.ECDSA) which becomes a subordinate CA. The KA then certifies the last certificate (CERT.OCE.ECKA) which contains the public key of the OCE used for key agreement.

In the same way, the CA-KLCC certifies the KA certificate (CERT.KA-KLCC.ECDSA) which becomes a subordinate CA. The KA then certifies the last certificate (CERT.SD.ECKA) which contains the public key of the SD used for key agreement.

Figure 3-2: Certificate Chain



4 SECURE CHANNEL PROTOCOL USAGE

This section describes the interaction that shall be implemented between an OCE (Off-Card Entity) and an SD (on-card entity) in order to establish an SCP11 session. An Application may implement the same interaction with the OCE by delegating the processing of protocol related APDU commands to its associated SD (see section 4.7).

4.1 Protocol Overview

Before setting up a secure channel, the OCE has to retrieve the SD's certificate, as shown below. The OCE may store the certificate (or parts of it) for use in future secure channel sessions. The OCE may also be provided with the certificate by some other means. In particular, in the case of SCP11c the OCE can retrieve the SD certificate(s) from the issuer(s) in advance through another channel out of scope of this document.

Figure 4-1: Initial Certificate Retrieval

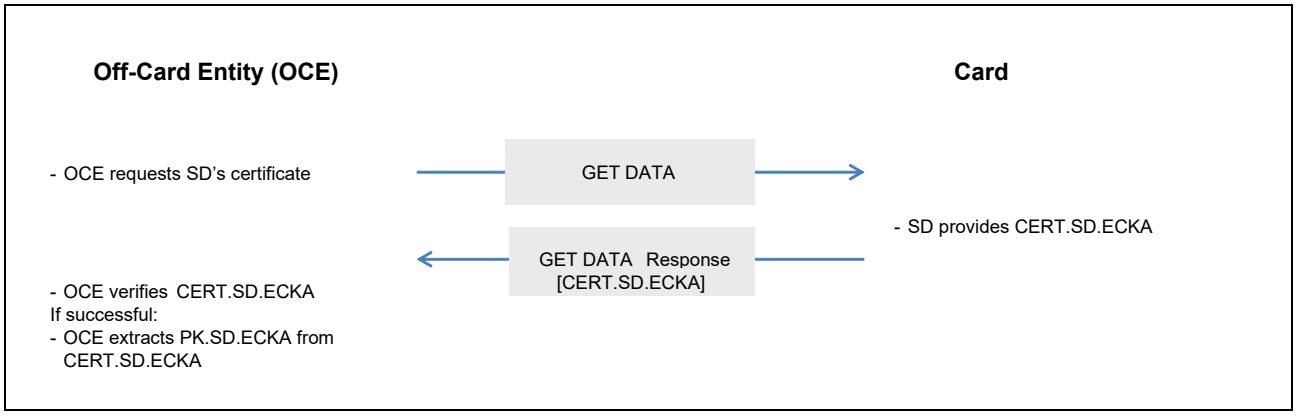


Figure 4-2, Figure 4-3, and Figure 4-4 provide an overview of the three variants of SCP11.

SCP11a and SCP11c provide mutual authentication of the OCE and the SD. For this purpose, the OCE has to provide the SD with its certificate in a PERFORM SECURITY OPERATION command prior to the establishment of the secure channel. For SCP11a, depending on the implementation option (see section 4.2), the SD may store the OCE public key extracted from the certificate persistently.

- If the SD stores the OCE public key persistently, it is not required for a PERFORM SECURITY OPERATION command to be sent immediately before the MUTUAL AUTHENTICATE command. The public key can be used in multiple future secure channel sessions, even in new card sessions after a power down.
- If the SD does not store the OCE public key persistently, the MUTUAL AUTHENTICATE command shall always be immediately preceded by a PERFORM SECURITY OPERATION command. In particular, the OCE public key shall be discarded in case of power down or termination of an SCP11 session.

As SCP11b authenticates the SD to the OCE but not vice versa, the PERFORM SECURITY OPERATION is not required.

SCP11a uses ephemeral keys for providing forward secrecy, while SCP11c does not use an ephemeral key on the SD side. Therefore, SCP11c can be used in a static command script, but it does not provide forward secrecy for the SD.

Figure 4-2: SCP11a Protocol Overview

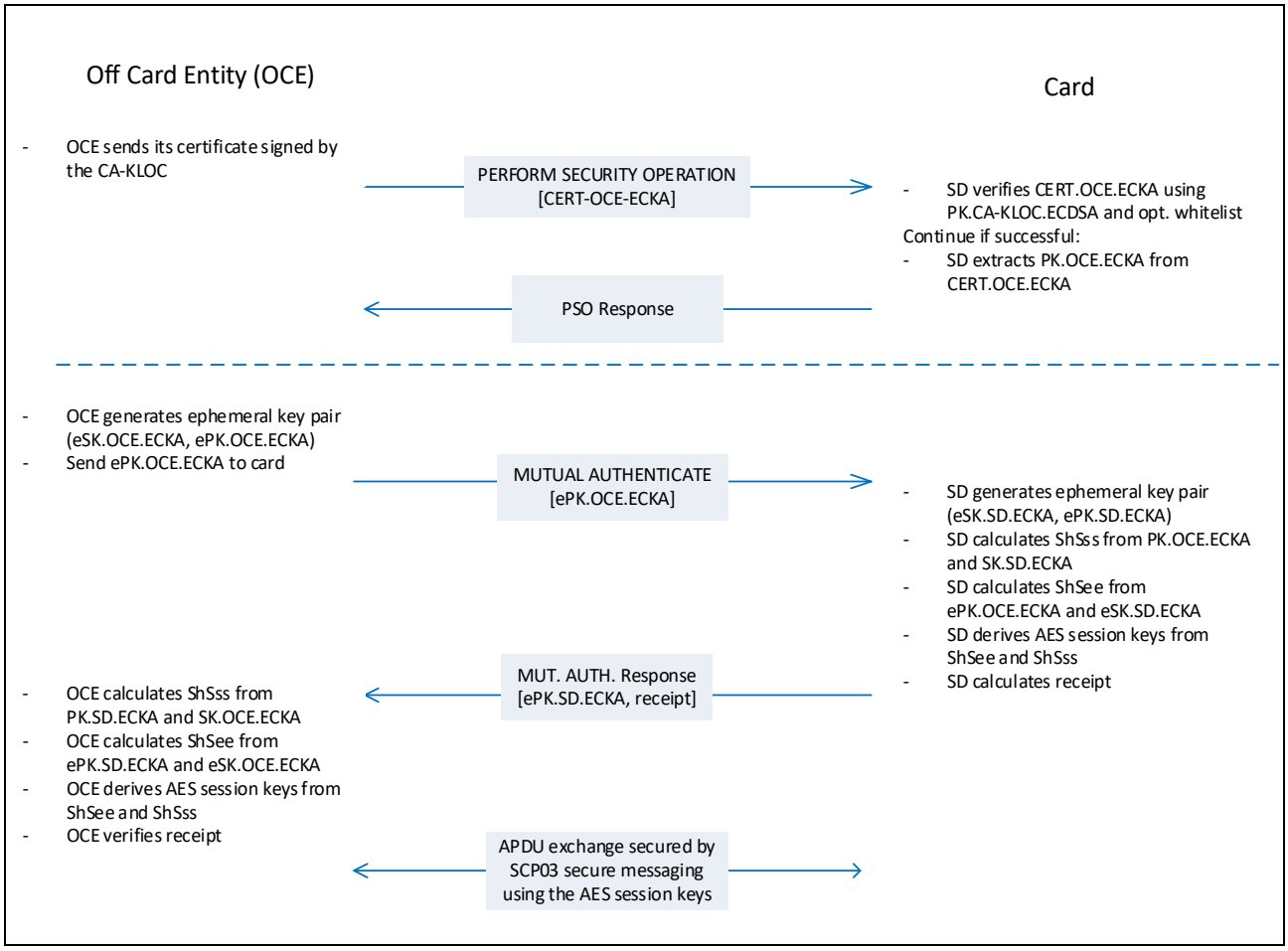


Figure 4-3: SCP11b Protocol Overview

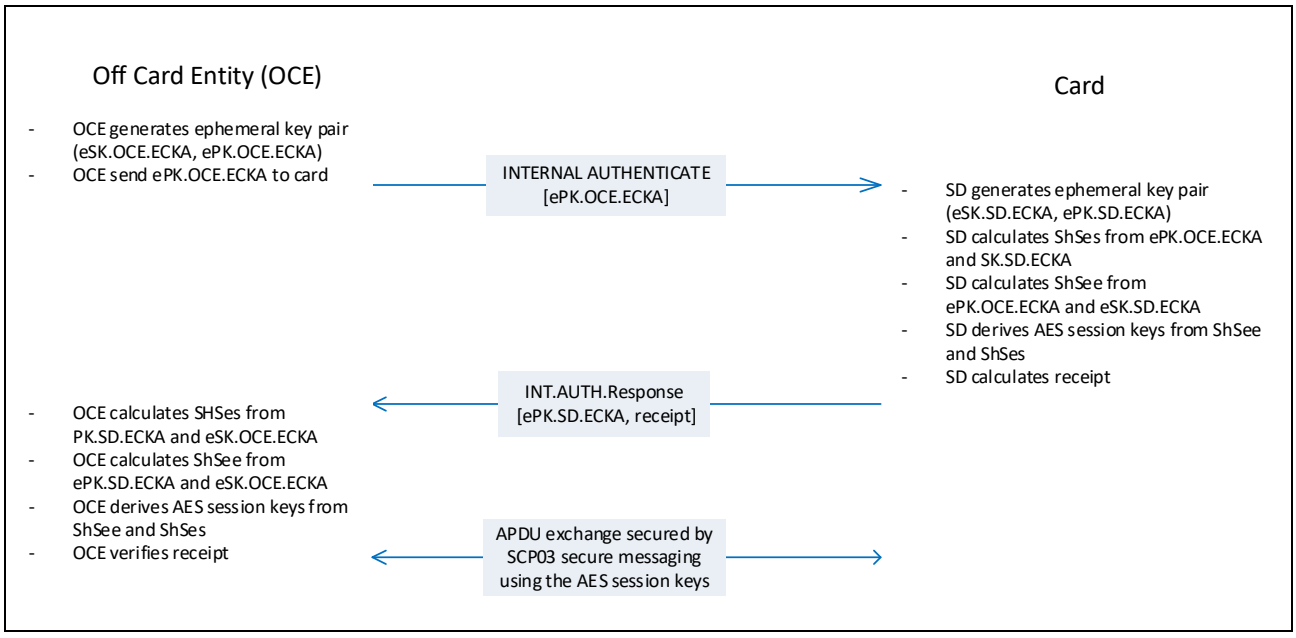
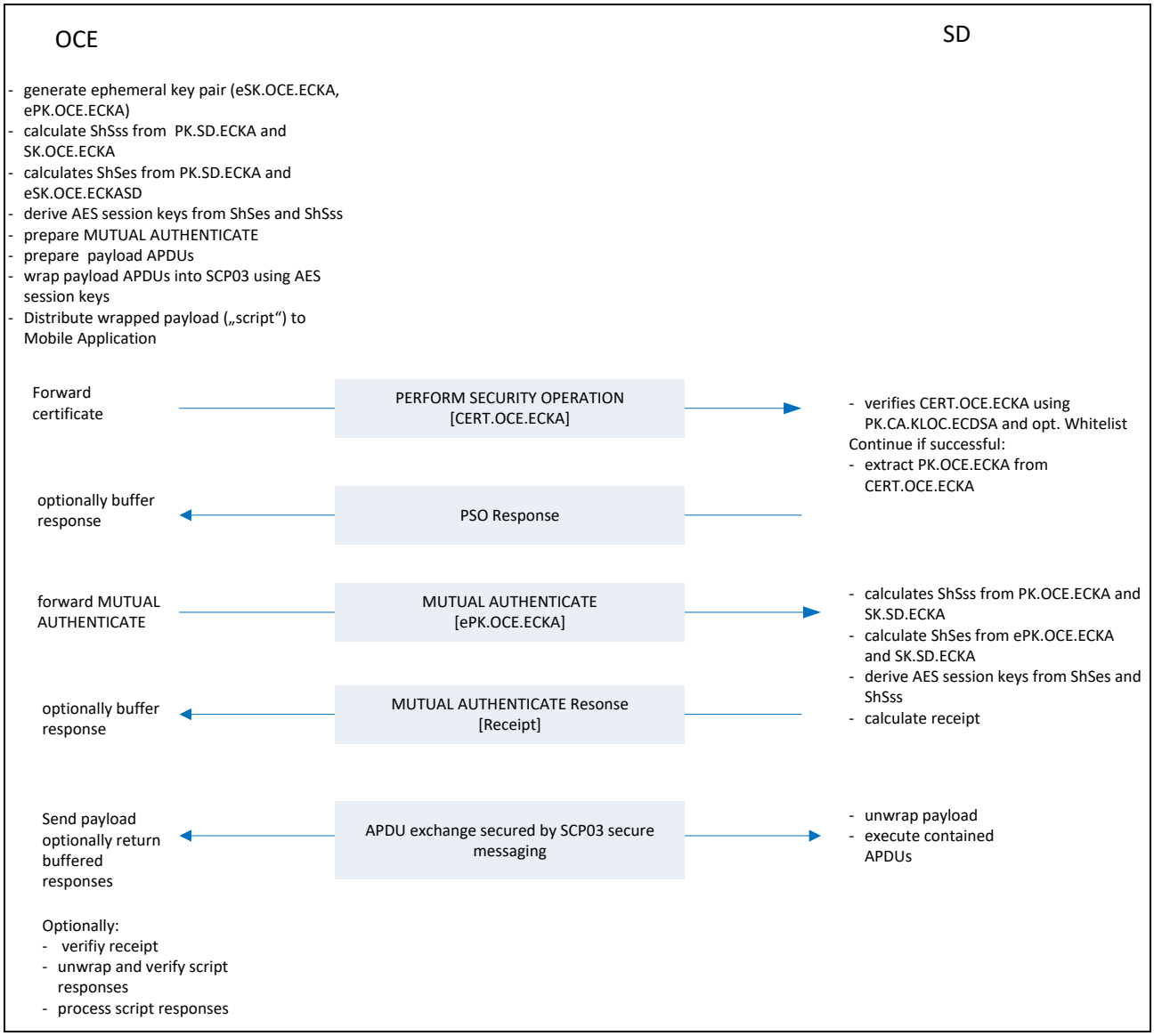


Figure 4-4: SCP11c Protocol Overview



The OCE may be a remote entity or may be split into a remote entity that precomputes data and a local agent (e.g. a mobile application on a smartphone) that forwards data to the SD. The OCE may precompute the commands or receive precomputed commands from another entity; this is out of scope of this document. After the execution of the APDU commands has finished, the local agent may return response data to the remote entity if required.

> ***Note:*** *If an OCE wishes to support several different Card Group IDs, e.g. because it wishes to support different issuers, it has to precompute several scripts using different CERT.SD.ECKA. In this case, the OCE may have to obtain several CERT.SD.ECKA from different vendors in order to precompute these scripts. The distribution infrastructure of these certificates is out of scope of this document.*

4.2 Secure Communication Configuration

Three levels of security are supported by SCP11:

- Authentication: Assurance that the peer entity is in fact the entity it claims to be
- Integrity and data origin authentication
- Confidentiality

Details for SCP11a, SCP11b, and SCP11c are given in section 4.3 and section 4.4.

In SCP11, implementation option “i” is formed as a bitmap on one or two bytes as follows:

Table 4-1: Values of Parameter “i” for byte 1

b8	b7	b6	b5	b4	b3	b2	b1	Description
							X	1: SCP11a supported
						X		1: SCP11b supported
					X			1: SD persistently stores PK.OCE.ECKA
				X				1: Certificate chain supported
			X					1: SCP11c supported
		X						1: SCP11c authorization mechanism supported ('BF20')
	X							0: Secure messaging using S8 mode (see [Amd D]) 1: Secure messaging using S16 mode (see [Amd D])
X								see note below 0: second byte is not present 1: second byte is present

Table 4-2: Values of Parameter “i” for byte 2

b8	b7	b6	b5	b4	b3	b2	b1	Description
							X	1: SCP11a authorization mechanism supported ('BF20')
						X		1: X.509 certificates supported
					X			1: GP legacy format certificates **not** supported
	X	X	X	X				RFU
X								Reserved; see note below

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

An implementation may support one or two or all variants of SCP11.

4.3 Authentication

Authentication is achieved through the process of initiating a Secure Channel and provides assurance to an entity that it is communicating with an authenticated entity.

For SCP11a and SCP11c only: The OCE authenticates itself to the SD by providing a certificate signed by the CA-KLOC and by providing the first APDU after secure channel establishment with a correct MAC. If an allowlist with one or more Certificate Serial Number entries exists in the SD for the CA-KLOC's public key, the SD also verifies that the certificate is contained in an allowlist. Else the SD accepts all certificates signed by the CA-KLOC.

Use of the allowlist can provide the following benefits:

- A strong binding to one (or multiple) OCE(s)
- Protection against compromised OCEs

It is recommended to use the allowlist also as a revocation mechanism for OCE certificates. However, if the allowlist is used in this way, special care shall be taken never to empty/remove the allowlist (i.e. if created, the allowlist shall always contain at least one certificate) because no restrictions apply (i.e. all certificates are accepted) once an allowlist is removed.

The SD authenticates to the OCE by providing a certificate signed by the CA-KLCC and by generating a receipt at the end of the key establishment procedure. Implementation of a revocation mechanism for the SD's certificate is recommended (e.g. by the OCE using an allowlist or a denylist of the SD certificates), but out of scope of this specification. Such a mechanism can protect the system against compromised SD keys.

SCP11a and SCP11c provide Mutual Authentication between the OCE and the SD.

SCP11b provides Authentication of the SD to the OCE only.

Applications can retrieve the current Security Level via the API method *getSecurityLevel()* to find out which variant is used during a session. A current Security Level of AUTHENTICATED or ANY_AUTHENTICATED indicates that mutual authentication was successful and that SCP11a or SCP11c is currently in use (see section 4.7).

> ***Note:*** *When executing the MUTUAL AUTHENTICATE command, the SD will be able to distinguish between SCP11a and SCP11c by evaluating tag '90' in the command data.*

4.4 Message Integrity and Data Confidentiality

Message Integrity and Data Confidentiality is achieved by secure messaging as defined in [Amd D], which is applied to all APDUs following the MUTUAL or INTERNAL AUTHENTICATE command.

Changing of the Security Level with the commands BEGIN or END R-MAC SESSION defined in [Amd D] shall not be supported by SCP11.

4.5 Forward Secrecy

Forward Secrecy assures the continued confidentiality of the data exchanged in a session even if the static private keys are compromised at a later point in time.

SCP11a and SCP11b provide Forward Secrecy (sometimes also called Perfect Forward Secrecy).

This is achieved by the ephemeral key pairs generated by the OCE and the SD which are used only once for the establishment of the session keys and which are destroyed immediately thereafter.

SCP11c does not provide Forward Secrecy because only one ephemeral key pair is used (generated on the OCE side).

4.6 Session Replay

For SCP11a and SCP11b, session replay is not possible as new session keys are generated for each session and the session key generation process includes an ephemeral key pair generated by the target SD.

For SCP11c, session replay is possible as the randomness of session keys only depends on an ephemeral key pair generated by the OCE. Therefore, operations performed within an SCP11c session are controlled as described below:

- The usage of PUT KEY, DELETE [key(s)], and SET STATUS commands shall not be allowed. The STORE DATA command is allowed but should not be used to load keys out of a secure and controlled environment. For use cases where keys need to be set up, it is recommended to use scenario #1 or scenario #3 from [Amd A].
- If the Security Level is ANY_AUTHENTICATED, the usage of some APDU commands and the usage of TLVs/DGIs in the STORE DATA command are subject to authorizations. See Annex B for details.

Only one SCP11c session per SD is allowed at a given time.

4.7 API and Security Level

An SD supporting SCP11 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 SCP11 related commands.

When using SCP11a or SCP11c, the API method `SecureChannel.getSecurityLevel()` shall return a value as described in Table 4-3, depending on:

- Key Usage Qualifier requested by the MUTUAL AUTHENTICATE command
- Subject Identifier of the OCE

Table 4-3: Security Levels in SCP11a and SCP11c

Subject Identifier	Key Usage Qualifier	Security Level
Does not match owner of the Security Domain	'34'	ANY_AUTHENTICATED | C_MAC | R_MAC
	'3C'	ANY_AUTHENTICATED | C_MAC | C_DECRYPTION | R_MAC | R_ENCRYPTION
	'1C'	ANY_AUTHENTICATED | C_MAC | C_DECRYPTION
	'74'	ANY_AUTHENTICATED | C_MAC | C_DECRYPTION | R_MAC
Matches owner of the Security Domain	'34'	AUTHENTICATED | C_MAC | R_MAC
	'3C'	AUTHENTICATED | C_MAC | C_DECRYPTION | R_MAC | R_ENCRYPTION
	'1C'	AUTHENTICATED | C_MAC | C_DECRYPTION
	'74'	AUTHENTICATED | C_MAC | C_DECRYPTION | R_MAC

When using SCP11b, the API method `SecureChannel.getSecurityLevel()` shall return a value as described in Table 4-4, depending on the Key Usage Qualifier requested by the MUTUAL AUTHENTICATE command.

Table 4-4: Security Levels in SCP11b

Key Usage Qualifier	Security Level
'34'	C_MAC | R_MAC
'3C'	C_MAC | C_DECRYPTION | R_MAC | R_ENCRYPTION
'1C'	C_MAC | C_DECRYPTION
'74'	C_MAC | C_DECRYPTION | R_MAC

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

- PERFORM SECURITY OPERATION
- MUTUAL AUTHENTICATE
- INTERNAL AUTHENTICATE
- GET DATA (without secure messaging), to retrieve data objects specific to the SCP11 protocol (see sections 7.2 and 7.3). 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 7.2.1)
  - 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.

A call to `decryptData()` or `encryptData()` shall throw an `ISOException` with reason code '6985' if Key-DEK is personalized, but not available for the calling application due to the settings in the Key Access Coding (see section 5.3.3). The secure channel session shall not be aborted.

The `SecureChannelx` interface is not supported (i.e. it is not possible to change the security level during the Secure Channel session). Support of the `SecureChannelx2` interface is optional.

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.
- The Current Security Level established in a Secure Channel Session is a bitmap combination of the following values: AUTHENTICATED, ANY_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;
  - After successful processing of a MUTUAL AUTHENTICATE command (for SCP11a and SCP11c):
    - Either AUTHENTICATED or ANY_AUTHENTICATED; and
    - C_MAC, R_MAC, C_DECRYPTION, and/or R_ENCRYPTION depending on the key usage qualifier specified in the MUTUAL AUTHENTICATE command (see Table 7-15).

- After successful processing of an INTERNAL AUTHENTICATE command (for SCP11b), C_MAC, R_MAC, C_DECRYPTION, and/or R_ENCRYPTION depending on the key usage qualifier specified in the INTERNAL AUTHENTICATE command (see Table 7-21).

Note: *The key usage qualifier contained in the command data of the MUTUAL or INTERNAL AUTHENTICATE command is used to determine the Security Level of the secure channel session.*

Note: *For SCP11c, card content management operations which are listed in the certificate are allowed in addition to those granted by the Security Level. For details refer to Annex B.*

As defined in [GPCS] section 10.4.2, ANY_AUTHENTICATED is the Security Level achieved if any OCE not being the owner of the SD authenticates using asymmetric cryptography. AUTHENTICATED is achieved if the owner of the SD or Application authenticates. The SD identifies the owner by the Subject Identifier (tag '5F20' for GP Certificates or 'id-sd-subject-id' extension for X.509 Certificates) in the OCE certificate matching the Application Provider Identifier of the SD or Application, which was provided as a parameter (TLV with tag '5F20' within the CRT TLV with tag 'B6') in the INSTALL [for install] command.

Note: *The CRT (tag 'B6') containing the Application Provider Identifier (tag '5F20') serves two purposes: To provide the Application Provider Identifier to the SD to be used in secure channel protocols with asymmetric cryptography and to provide token information for Delegated Management.*

4.8 Protocol Rules

In accordance with the general rules described in [GPCS] section 10, the following protocol rules apply to SCP11:

- The successful initiation of a Secure Channel Session shall set the Current Security Level to the Security Level indicated in the MUTUAL or INTERNAL AUTHENTICATE command.
- The Current Security Level shall apply to the entire Secure Channel Session.
- 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 different from NO_SECURITY_LEVEL), 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 does not match 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 on any of the following:
  - Attempt to initiate a new Secure Channel Session
  - Termination of the Application Session (e.g. new Application selection)
  - 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 ECC Keys

Table 5-1: ECC Keys

Key	Usage	Length	Remark
eSK.SD.ECKA	Ephemeral private key of the SD used for key agreement	see below	SCP11a and SCP11b only
ePK.SD.ECKA	Ephemeral public key of the SD used for key agreement	see below	SCP11a and SCP11b only
eSK.OCE.ECKA	Ephemeral private key of the OCE used for key agreement	see below	
ePK.OCE.ECKA	Ephemeral public key of the OCE used for key agreement	see below	
SK.SD.ECKA	Private key of the SD used for key agreement	see below	
PK.SD.ECKA	Public key of the SD used for key agreement	see below	
CERT.SD.ECKA	Certificate containing the public key of the SD used for key agreement, signed by the CA-KLCC (or subordinate KA-KLCC)	see below	
SK.OCE.ECKA	Private key of the OCE used for key agreement	see below	SCP11a and SCP11c only
PK.OCE.ECKA	Public key of the OCE used for key agreement	see below	SCP11a and SCP11c only
CERT.OCE.ECKA	Certificate containing the public key of the OCE used for key agreement, signed by the CA-KLOC (or subordinate KA-KLOC)	see below	SCP11a and SCP11c only
SK.CA-KLOC.ECDSA	Private key of the CA-KLOC used for signing certificates	see below	SCP11a and SCP11c only
PK.CA-KLOC.ECDSA	Public key of the CA-KLOC used for verifying certificates	see below	SCP11a and SCP11c only
CERT.CA-KLOC.ECDSA	Certificate containing the public key of the CA-KLOC used for verifying certificates, self-signed	see below	SCP11a and SCP11c only
CERT.KA-KLOC.ECDSA	Certificate containing the public key of the KA-KLOC used for verifying certificates, signed by the CA-KLOC (or another intermediate KA-KLOC)	see below	SCP11a and SCP11c only
SK.KA-KLOC.ECDSA	Private key of the KA-KLOC used for signing certificates	see below	SCP11a and SCP11c only
PK.KA-KLOC.ECDSA	Public key of the KA-KLOC used for verifying certificates	see below	SCP11a and SCP11c only

Key	Usage	Length	Remark
PK.CA-KLCC.ECDSA	Public key of the CA-KLCC used for verifying certificates	see below	
CERT.CA-KLCC.ECDSA	Certificate containing the public key of the CA-KLCC used for verifying certificates, self-signed	see below	
CERT.KA-KLCC.ECDSA	Certificate containing the public key of the KA-KLCC used for verifying certificates, signed by the CA-KLCC (or another intermediate KA-KLCC)	see below	
SK.KA-KLCC.ECDSA	Private key of the KA-KLCC used for signing certificates	see below	
PK.KA-KLCC.ECDSA	Public key of the KA-KLCC used for verifying certificates	see below	

All ECC keys shall reference the same curve parameters. Thus all keys have the same length. The curve parameters shall be available on the SD prior to any SCP11 related operation.

It is recommended to use one of the standardized curves described in [GPCS] section B.4.1.

SK.SD.ECKA and PK.CA-KLOC.ECDSA are keys stored in the SD supporting SCP11, each with its own unique combination of Key Identifier and Key Version Number. Several of these keys may be stored in an SD.

The following Key Identifiers (KID) shall be used:

- KID '10' for PK.CA-KLOC.ECDSA
- KID '11' for SK.SD.ECKA used for SCP11a
- KID '12' for the optional static Key-DEK used with SCP11a only
- KID '13' for SK.SD.ECKA used for SCP11b
- KID '14' for the optional static Key-DEK used with SCP11b only
- KID '15' for SK.SD.ECKA used for SCP11c
- KID '16' for the optional static Key-DEK used with SCP11c only
- KID from '20' to '2F' for additional PK.CA-KLOC.ECDSA

  Note: *Even if KID values are fixed, they are also provided in the commands defined in section 6. This allows reusing the commands for other purposes in the future.*

  Note: *Assignments of Key Version Numbers (KVN) may be defined in configurations.*

A related pair of SK.SD.ECKA and Key-DEK shall have the same KVN.

CERT.SD.ECKA is a data object stored in the SD supporting SCP11, referencing the Key Identifier and Key Version Number of the associated SK.SD.ECKA (see section 7.8). Each SK.SD.ECKA must have one associated CERT.SD.ECKA.

If certificate chaining and one or more Key Authorities are used, the public key extracted from each intermediary certificate should not be stored persistently but shall be used to verify the next certificate in the chain down to PK.OCE or PK.SD.

Within the same SD, different PK.CA-KLOC.ECDSA may be stored and used to verify CERT.OCE.ECKA, or the first CERT.KA-KLOC.ECDSA of a certificate chain if certificate chaining is used. A range of Key Identifiers is defined for storing different PK.CA-KLOC.ECDSA.

When contained in a command or a response, static or ephemeral public keys shall be formatted using uncompressed encoding as specified in [TR 03111] section 3.1.1, with most significant byte coming first (hence the value shall start with the coding identifier byte '04'). Thus each key value field will have a fixed length of twice the order length plus one. For ephemeral public keys, this key value field is the data field of the TLV with tag '5F49'.

Note: *The ephemeral private keys and the shared secrets ShSee, ShSss, ShSes, ShSse, and ShS are as sensitive as the static private keys and need to be protected accordingly.*

5.2 AES Keys

Table 5-2: Security Domain Secure Channel Keys

Key	Usage	Length	Remark
Data Encryption Key (Key-DEK)	Sensitive Data Encryption and Decryption (AES)	see below	Optional
Session Data Encryption Key (S-DEK)	Sensitive Data Encryption and Decryption (AES)	see below	Conditional / Dynamically
Secure Channel Session Encryption Key (S-ENC)	Used for data confidentiality	see below	Dynamically
Secure Channel Session Message Authentication Code Key for Command (S-MAC)	Used for data and protocol integrity	see below	Dynamically
Secure Channel Session Message Authentication Code Key for Response (S-RMAC)	Used for data and protocol integrity	see below	Dynamically

See section 5.1 for KID values for the Key-DEK.

If no static Key-DEK is present in the SD, a session DEK (S-DEK) will be generated together with the other session keys. It will be used for sensitive data encryption and decryption during the secure channel session instead of a Key-DEK.

The recommended length of these AES keys depends on the length of the ECC keys according to the following table.

Table 5-3: Recommended Length of AES Keys

ECC Key Length in Bits	Recommended Length of AES Keys in Bits
256-383	128
384-511	192
512+	256

Note: *To provide a balanced security, it is strongly recommended to implement this pairing. However, an implementation may also choose to tolerate other combinations. The security implications have to be considered carefully. For example, if ECC 256 is used to establish AES256 keys, these AES keys cannot be considered to provide their full strength.*

Note: *Although SCP11 uses the secure messaging mechanisms of SCP03, an SD may support SCP03 with static keys as specified in [Amd D] independently. Support for SCP11 does not imply support for SCP03 nor affect any configuration settings for SCP03.*

5.3 Cryptographic Usage

5.3.1 AES Session Keys

AES session keys shall be generated every time a Secure Channel is initiated and are used for secure messaging on subsequent commands.

Session keys are generated to ensure that a different set of keys is used for each Secure Channel session.

5.3.2 Secure Messaging

Secure Messaging as defined in [Amd D] shall be applied to all commands following a successful MUTUAL or INTERNAL AUTHENTICATE command.

For SCP11a and SCP11b, only two Security Levels for Secure Messaging are defined in this specification; the Security Level is set in the key usage qualifier data object of the MUTUAL or INTERNAL AUTHENTICATE command:

- C-MAC and R-MAC only
- C-DECRYPTION, R-ENCRYPTION, C-MAC, and R-MAC

For SCP11c, four Security Levels for Secure Messaging are defined in this specification; the Security Level is set in the key usage qualifier data object of the MUTUAL AUTHENTICATE command:

- C-MAC and R-MAC only
- C-DECRYPTION, R-ENCRYPTION, C-MAC, and R-MAC
- C-DECRYPTION and C-MAC
- C-DECRYPTION, C-MAC, and R-MAC

The MAC chaining value of the first APDU command after the MUTUAL or INTERNAL AUTHENTICATE command shall be set to the value of the receipt returned by the SD in the MUTUAL or INTERNAL AUTHENTICATE response.

> ***Note:*** *When using SCP03, the first MAC is calculated on the EXTERNAL AUTHENTICATE command and the following commands are bound to the secure channel initiation via the MAC chaining. As the equivalent in SCP11 for this first MAC is the receipt, it is used as the first MAC chaining value to bind the following commands to the SCP11 secure channel initiation.*

5.3.3 Key Access Conditions

The Key Access Conditions as defined in [GPCS] shall be supported by the SD for SK.SD.ECKA and Key-DEK.

Its value shall be interpreted as follows:

- Setting up a Secure Channel with an SK.SD.ECKA having Key Access Conditions set to '00' shall always be accepted.
- Setting up a Secure Channel with an SK.SD.ECKA having Key Access Conditions set to '01' shall only be accepted if the SD holding the keys is the selected or the targeted application.
- Setting up a Secure Channel with an SK.SD.ECKA having Key Access Conditions set to '02' shall only be accepted if an application associated with the SD holding the keys is the selected or the targeted application. If the associated application is an SD, additional requirements may apply (e.g. the SD not having a key set of its own).

  Note: *The setting '02' for SK.SD.ECKA is useful to prevent a secure channel from being established by the SD and being used for application management or key update.*

- An attempt by the SD to use the Key-DEK having Key Access Conditions set to '02' shall fail. It shall be allowed for Key Access Conditions set to '00' or '01'.
- An attempt by an application (including SDs) associated with the SD to use the Key-DEK having Key Access Conditions set to '01' shall fail. It shall be allowed for Key Access Conditions set to '00' or '02'.

6 CERTIFICATES

This section introduces the two formats than can be supported in commands (see section 7). All certificates in a certificate chain shall be encoded in the same format.

6.1 GP Certificates

This format is supported by default.

Table 6-1: Certificate Format

Tag	Length	Value Description			MOC
'7F21'	Var	Certificate			M
		Tag	**Length**	**Value Description**	**MOC**
		'93'	1-16	Certificate Serial Number	M
		'42'	1-16	CA Identifier (CA-KLOC, CA-KLCC, KA-KLOC or KA-KLCC Identifier)	M
		'5F20'	Var	Subject Identifier	M
		'95'	1-2	Key Usage: • '82': Digital signature verification (CERT.KA-KLOC.ECDSA or CERT.KA-KLCC.ECDSA) • '00 80': Key agreement (CERT.OCE.ECKA or CERT.SD.ECKA)	M
		'5F25'	4	Effective Date (YYYYMMDD, BCD format)	O
		'5F24'	4	Expiration Date (YYYYMMDD, BCD format)	M
		'53' or '73'	1-127	Discretionary Data	O
		'BF20'	Var	Authorizations under SCP11a or SCP11c	O
		'7F49'	Var	Public Key – for details, see tables below	M
		'5F37'	Var	Signature	M

Tags '53' and '73' are mutually exclusive.

The TLV-encoded data listed in the following table are signed off-card with SK.CA-KLOC.ECDSA (resp. SK.CA-KLCC.ECDSA) for the first CERT.KA-KLOC.ECDSA (resp. CERT.KA-KLCC.ECDSA), if any, or SK.KA-KLOC.ECDSA (resp. SK.KA-KLCC.ECDSA) for the next certificate, or private key associated with the previous certificate for CERT.OCE.ECKA (resp. CERT.SD.ECKA). The signature is stored in the content of tag '5F37' (signature), as described in [Amd A].

Note: CERT.SD.ECKA is identical to the format defined for CERT.CASD.ECKA in [Amd A]. Tag '42' identifies the owner of the SD; tag '45' identifies the Security Domain Image Number.

Table 6-2: Data Signed to Generate the Certificate

Tag	Length	Value Description	MOC
'93'	1-16	Certificate Serial Number	M
'42'	1-16	CA Identifier (CA-KLOC, CA-KLCC, KA-KLOC or KA-KLCC Identifier)	M
'5F20'	Var	Subject Identifier	M
'95'	1-2	Key Usage: • '82': Digital signature verification (CERT.KA-KLOC.ECDSA or CERT.KA-KLCC.ECDSA) • '00 80': Key agreement (CERT.OCE.ECKA or CERT.SD.ECKA)	M
'5F25'	4	Effective Date (YYYYMMDD, BCD format)	C
'5F24'	4	Expiration Date (YYYYMMDD, BCD format)	M
'53' or '73'	1-127	Discretionary Data	C
'BF20'	Var	Authorizations under SCP11c or SCP11a	C
'7F49'	Var	Public Key (see Table 6-3)	M

Tags '5F25', '53', '73', and 'BF20' shall be included in signed data if they are present in the OCE certificate.

The Public Key Data Object contains an Elliptic Curves (EC) public key and the corresponding key parameter reference.

Table 6-3: Public Key structure

Tag	Length	Value Description	MOC
'B0'	Var	Public key – Q	M
'F0'	1 or 2	Key Parameter Reference	M

6.2 X.509 Certificates

This section describes the X.509 certificate format. This format can only be used if supported by the Secure Channel Configuration (see section 4.2)

Those certificates SHALL follow [RFC 5280], with the specific coding given in this section.

In particular:

- 'Issuer' and 'Subject' fields SHALL be limited to standard attributes defined in ITU-T X.520 [ITU X520] and RFC 4519 [RFC 4519].
- Certificates SHALL contain all extensions defined in their respective profile, except if stated otherwise.
- Certificates SHALL NOT contain the `freshestCRL` extension (use of Delta CRL is not supported).
- The Subject Key Identifier SHOULD be computed using method 1 specified in section 4.2.1.2 in [RFC 5280] for all the certificates.

Certificates are described using table representation for easiness but conform to the ASN.1 format given in [RFC 5280].

6.2.1 Certificates Description

6.2.1.1 Algorithm Identifiers and Parameters

This section provides the values to be set in 'AlgorithmIdentifier.algorithm' and 'AlgorithmIdentifier.parameters' fields of the certificate for each of the algorithms used in this specification.

For section 'subjectPublicKeyInfo' the following settings SHALL apply:

'AlgorithmIdentifier.algorithm' field SHALL be set to: "iso(1) member-body(2) us(840) ansi-X9-62(10045) keyType(2) ecPublicKey(1)" as defined in [RFC 5480].

'AlgorithmIdentifier.parameters' field SHALL be set to one of the following values:

- As defined in [RFC 5639]:
  - For BrainpoolP256r1: "iso(1) identified-organization(3) teletrust(36) algorithm(3) signatureAlgorithm(3) ecSign(2) ecStdCurvesAndGeneration(8) ellipticCurve(1) versionOne(1) brainpoolP256r1(7)"
  - For BrainpoolP384r1: "iso(1) identified-organization(3) teletrust(36) algorithm(3) signatureAlgorithm(3) ecSign(2) ecStdCurvesAndGeneration(8) ellipticCurve(1) versionOne(1) brainpoolP384r1(11)"
  - For BrainpoolP512r1: "iso(1) identified-organization(3) teletrust(36) algorithm(3) signatureAlgorithm(3) ecSign(2) ecStdCurvesAndGeneration(8) ellipticCurve(1) versionOne(1) brainpoolP512r1(13)"
- As defined in [RFC 5480]:
  - For NIST P-256: "iso(1) member-body(2) us(840) ansi-X9-62(10045) curves(3) prime(1) secp256r1(7)"
  - For NIST P-384: " iso(1) identified-organization(3) certicom(132) curve(0) secp384r1(34)"
  - For NIST P-521: " iso(1) identified-organization(3) certicom(132) curve(0) secp521r1(35)"

For sections 'signature' and 'signatureAlgorithm' the following settings SHALL apply:

- As defined in [RFC 5758]:
  - 'AlgorithmIdentifier.algorithm' field SHALL be set to one of the following values, based on rules described in [GPCS] section B.4.3:
    - ECDSA with SHA-256: "iso(1) member-body(2) us(840) ansi-X9-62(10045) signatures(4) ecdsa-with-SHA2(3) ecdsa-with-SHA256(2)"
    - ECDSA with SHA-384: "iso(1) member-body(2) us(840) ansi-X9-62(10045) signatures(4) ecdsa-with-SHA2(3) ecdsa-with-SHA384(3)"
    - ECDSA with SHA-512: "iso(1) member-body(2) us(840) ansi-X9-62(10045) signatures(4) ecdsa-with-SHA2(3) ecdsa-with-SHA512(4)"
  - 'AlgorithmIdentifier.parameters' field SHALL be omitted as defined in [RFC 5758] section 3.2.

6.2.1.2 Certificates Common Fields

The table below describes the common fields that all certificates defined in this section SHALL contain. A certificate MAY have additional fields or a different content for a field; in that case it will be indicated in its specific description.

Table 6-4: X.509 Certificate Common Fields

Certificate Common Fields		
Field	**Value Description**	
tbsCertificate	Data to be signed	
	version	Version SHALL be 3 (value is 2) as extensions are used in all certificates defined in this specification.
	serialNumber	Certificate serial number.
	signature	Contains the algorithm identifier used by the issuer to compute the value of the field 'signatureValue'. NOTE: The algorithm identifier value SHALL be the same as the one of the field 'signatureAlgorithm'.
	issuer	Distinguished Name of the entity that has signed the certificate. It SHALL match the 'subject' field of the issuing entity certificate.
	validity	Validity period of the certificate
	subject	Distinguished Name of the entity owning the certificate. Example of DN: cn = GP CA ou = GP Trust Network o = GP c = UK
	subjectPublicKeyInfo	Contains the algorithm identifier, parameters and public key value. Algorithm identifier and parameters SHALL be set according to section 6.2.1.1. subjectPublicKeyInfo.subjectPublicKey contains the public key value and SHALL be coded as defined in [RFC 5480].
	extensions	Extension for Subject Key Identifier. ([RFC 5280] section 4.2.1.2). This extension SHALL be set with: extnID = id-ce-subjectKeyIdentifier critical = false extnValue = <Identifier of the public key bound in the certificate>
signatureAlgorithm	See section 6.2.1.1.	
signatureValue	Signature value	

6.2.1.3 CA-KLOC and CA-KLCC Certificate

The table below describes the specific fields of a CERT.CA-KLOC.ECDSA or CERT.CA-KLCC.ECDSA in complement of the description given in section 6.2.1.2:

Table 6-5: CA-KLOC and CA-KLCC Certificate fields

CERT.CA-KLOC.ECDSA or CERT.CA-KLCC.ECDSA		MOC
Field	**Value Description**	
issuer	**This SHALL be identical to 'subject' field value.**	**M**
Extension for Key usage (RFC 5280 section 4.2.1.3)	extnID = id-ce-keyUsage critical = true extnValue = { keyCertSign (5), cRLSign(6) } -- cRLSign is optional	**M**
Extension for Certificate Policies (RFC 5280 section 4.2.1.4)	extnID = id-ce-certificatePolicies critical = true extnValue = id-kloc-ca or id-klcc-ca (see Annex D)	**M**
Extension for Basic Constraints (RFC 5280 section 4.2.1.9)	extnID = id-ce-basicConstraints critical = true extnValue = { cA = true }	**M**
Extension for subjectAltName (RFC 5280 section 4.2.1.6)	extnID = id-ce-subjectAltName critical = false extnValue = { registeredID (8) = CA OID }	**O**

6.2.1.4 KA-KLOC and KA-KLCC Certificate

The table below describes the specific fields of a CERT.KA-KLOC.ECDSA or CERT.KA-KLCC.ECDSA in complement of the description given in section 6.2.1.2:

Table 6-6: KA-KLOC and KA-KLCC Certificate fields

CERT.KA-KLOC.ECDSA or CERT.KA-KLCC.ECDSA		MOC
Field	**Value Description**	
Extension for Authority Key Identifier (RFC 5280 [17] section 4.2.1.1)	extnID = id-ce- authorityKeyIdentifier critical = false extnValue = <Identifier of the public key to verify this certificate>	M
Extension for Key usage (RFC 5280 section 4.2.1.3)	extnID = id-ce-keyUsage critical = true extnValue = { keyCertSign (5), cRLSign(6) } -- cRLSign is optional	M
Extension for Certificate Policies (RFC 5280 section 4.2.1.4)	extnID = id-ce-certificatePolicies critical = true extnValue = id-kloc-ka or id-klcc-ka (see Annex D)	M
Extension for Basic Constraints (RFC 5280 section 4.2.1.9)	extnID = id-ce-basicConstraints critical = true extnValue = { cA = true }	M
Extension for subjectAltName (RFC 5280 section 4.2.1.6)	extnID = id-ce-subjectAltName critical = false extnValue = { registeredID (8) = KA OID }	O

6.2.1.5 OCE Certificate

The table below describes the specific fields of a CERT.OCE.ECKA in complement of the description given in section 6.2.1.2:

Table 6-7: OCE Certificate fields

CERT.OCE.ECKA		MOC
Field	**Value Description**	
Extension for Authority Key Identifier (RFC 5280 [17] section 4.2.1.1)	extnID = id-ce- authorityKeyIdentifier critical = false extnValue = <Identifier of the public key to verify this certificate>	**M**
Extension for Key usage (RFC 5280 [17] section 4.2.1.3)	extnID = id-ce-keyUsage critical = true extnValue = keyAgreement (4)	**M**
Extension for Certificate Policies (RFC 5280 [17] section 4.2.1.4)	extnID = id-ce-certificatePolicies critical = true extnValue = id-oce (see Annex D)	**M**
Extension for subjectAltName (RFC 5280 [17] section 4.2.1.6)	extnID = id-ce-subjectAltName critical = false extnValue = { registeredID (8) = OCE OID }	**O**
GP Extension for SCP11 authorization	extnID = id-scp11-auth (see Annex D) critical = true extnValue = < OCTET STRING(value of tag BF20)>	**C**
GP Extension for subject id	extnID = id-sd-subject-id (see Annex D) critical = true extnValue = < OCTET STRING(value of tag 5F20)> If present, the value of this extension shall be compared to the value of tag '5F20' stored in the Security Domain. If not present, the value of the 'subject' field of the X509 cert (i.e. Distinguished Name) shall be compared instead to the value of tag '5F20' stored in the Security Domain. The result of this comparison is to be used to determine the security level of the SCP11 session (either AUTHENTICATED or ANY AUTHENTICATED). See section 4.7	**C**

6.2.1.6 SD Certificate

The table below describes the specific fields of a CERT.SD.ECKA in complement of the description given in section 6.2.1.2:

Table 6-8: SD Certificate fields

CERT.SD.ECKA		MOC
Field	**Value Description**	
subject	Distinguished Name of the SD. It SHALL include, at least, 'serialNumber' attributes. Other attributes MAY be included for information. The 'organization' attribute corresponds to tag 42 The 'serialNumber' attribute SHALL be the SDIN (corresponding to tag 45) as a hexadecimal PrintableString. Example of an SD DN: o = ACME serialNumber = 0102030405060708090A0B	**M**
Extension for Authority Key Identifier (RFC 5280 [17] section 4.2.1.1)	extnID = id-ce-authorityKeyIdentifier critical = false **extnValue = <Identifier of the public key to verify this certificate>**	**M**
Extension for Key usage (RFC 5280 [17] section 4.2.1.3)	extnID = id-ce-keyUsage critical = true extnValue = keyAgreement (4)	**M**
Extension for Certificate Policies (RFC 5280 [17] section 4.2.1.4)	extnID = id-ce-certificatePolicies critical = true extnValue = id-sd (see Annex D)	**M**
GP Extension for Discretionary Data	extnID = id-disc-data (see Annex D) critical = false extnValue = <OCTET STRING (either TLV with tag '53' or '73' found in legacy GP certificate format)>	**O**

7 COMMANDS

The following table presents the new commands involved in Secure Channel Initiation and in SD Personalization when SCP11a/b/c is used.

Table 7-1: SCP11 Command Support

Command	Used By
GET DATA (ECKA Certificate)	SCP11a, b, and c
PERFORM SECURITY OPERATION	SCP11a and c
MUTUAL AUTHENTICATE	SCP11a and c
INTERNAL AUTHENTICATE	SCP11b
STORE DATA (ECKA Certificate)	SCP11a, b, and c
STORE DATA (Allowlist)	SCP11a and c

Note: *STORE DATA for the key establishment scenarios in Card Specification Amendments A and E use data structures with the same CRT. However, these can be clearly distinguished as the scenarios use DGI format whereas SCP11 uses TLV format.*

7.1 General Coding Rules

7.1.1 SCP Identifier and Parameters

The value field of SCP identifier and parameters shall be coded as follows:

The SCP identifier (byte 1) shall be set to '11'.

The SCP parameters (byte 2) are defined as follows:

Table 7-2: Parameters for SCP11

b8	b7	b6	b5	b4	b3	b2-b1	Description
–	–	–	–	–	–	XX	00: Indicates SCP11b 01: Indicates SCP11a 11: Indicates SCP11c 10: RFU
–	–	–	–	–	X	–	0: Do not include Host and Card ID / Card Group ID in key derivation process 1: Include Host and Card ID / Card Group ID in key derivation process
X	X	X	X	X	–	–	RFU (0)

Note: *The use of b3 is aligned with the coding defined in [Amd A].*

Note: *SCP11a and SCP11b use Card ID, while SCP11c uses Card Group ID.*

7.2 GET DATA (Certificate Store) Command

The GET DATA command is defined in [GPCS] section 11.3.

It is used by the OCE to retrieve an SCP11 Certificate Store from the SD.

The SD shall support bit b8 of the class byte set to 1 – support for bit b8 set to 0 is optional; the instruction code shall be set to 'CA'. The parameters P1 and P2 shall be set to 'BF 21'.

The data field of the command message shall be coded according to the following table.

Table 7-3: Data Field of GET DATA (ECKA Certificate(s)) Command

Tag	Length	Value Description			MOC
'A6'	4	Control Reference Template (Key Agreement)			M
		Tag	**Length**	**Value Description**	**MOC**
		'83'	2	byte 1: Key Identifier byte 2: Key Version Number	M

The SD shall return the SCP11 Certificate Store linked to the private key SK.SD.ECKA which is referenced by the CRT.

7.2.1 Response Message

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

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.

The response data field shall be coded according to one of the formats described in the following sections.

Table 7-4: Data Field of GET DATA (ECKA Certificate(s)) Response

Tag	Length	Value Description	MOC
'BF 21'	Var	SCP11 certificate store	M
		Value Description	**MOC**
		CERT.KA-KLCC.ECDSA	O
		….	O
		CERT.SD.ECKA	M

7.2.2 Processing Rules

For each certificate, the OCE shall verify at least the following:

- The signature of the certificate, using:
  - PK.CA-KLCC.ECDSA for the 1st (or only) certificate
  - The public key extracted from the previous CERT.KA-KLCC.ECDSA, if certificate chaining is used
- The Expiration Date, to ensure that the certificate is still valid
- The correctness of the Key Usage (i.e. shall be '00 80' (ECKA) for the last (or only) certificate)

In addition, the OCE should check either an allowlist or a revocation list for the SD's certificates. Such an allowlist or a revocation list should be maintained off-card.

> ***Note:*** *The OCE can retrieve the Key Version Number and Key Identifier of the ECC keys available in the SD by retrieving the Key Information Template using a GET DATA command. A stored PK.OCE.ECKA is not included in the Key Information Template.*

7.3 GET DATA (CA-KLOC KID-KVN) Command

The GET DATA command is defined in [GPCS] section 11.3.

It is used by the OCE to retrieve from the SD the KID and the KVN corresponding to a given CA-KLOC Identifier.

The SD shall support bit b8 of the class byte set to 1 – support for bit b8 set to 0 is optional; the instruction code shall be set to 'CA'. The parameters P1 and P2 shall be set to '00 83'.

The data field of the command message shall be coded according to the following table.

Table 7-5: Data Field of GET DATA (CA-KLOC KID-KVN) Command

Tag	Length	Value Description			MOC
'A6'	Var	Control Reference Template			M
		Tag	**Length**	**Value Description**	**MOC**
		'42'	Var	CA-KLOC Identifier	M

The SD shall return the KID and KVN linked to the public key PK.CA-KLOC.ECDSA referenced by the CRT, encapsulated according to the following table.

Table 7-6: Data Field of GET DATA (CA-KLOC KID-KVN) Response

Tag	Length	Value Description	MOC
'83'	2	byte 1: Key Identifier byte 2: Key Version Number	M

7.4 GET DATA (Supported CA Identifiers) Command

The GET DATA command is defined in [GPCS] section 11.3.

It is used by the OCE to retrieve from the SD either all CA-KLOC or all CA-KLCC Identifiers.

The SD shall support bit b8 of the class byte set to 1 – support for bit b8 set to 0 is optional; the instruction code shall be set to 'CA'.

The parameters P1 and P2 shall be set to 'FF33' to retrieve CA-KLOC Identifiers and 'FF34' to retrieve CA-KLCC Identifiers. The data field of the command message shall be empty.

The SD shall return all CA-KLOC or CA-KLCC Identifiers accordingly together with related Key Identifiers and Key Version Numbers, as described in the following table.

Table 7-7: Data Field of GET DATA (Supported CA Identifiers) Response

Tag	Length	Value Description			MOC
'FF33' or 'FF34'	Var	Control Reference Template			M
		Tag	**Length**	**Value Description**	**MOC**
		'42'	Var	CA Identifier	C
		'83'	2	KID / KVN	C
		…	…	…	C
		'42'	Var	CA Identifier	C
		'83'	2	KID / KVN	C

For a CA-KLOC Identifier, the returned KID/KVN relates to a given PK.CA-KLOC.ECDSA. For a CA-KLCC Identifier, the returned KID/KVN relates to a given SK.SD.ECKA.

7.5 PERFORM SECURITY OPERATION Command

7.5.1 Definition and Scope

The PERFORM SECURITY OPERATION command is used to submit CERT.OCE.ECKA, or possibly a chain of certificates ending with CERT.OCE.ECKA. This is required as a precondition to the initiation of a SCP11a or SCP11c secure channel.

This command does not terminate an ongoing secure channel session.

For each certificate:

- The certificate's signature shall be verified.
- The Certificate Serial Number (CSN) shall be checked as further explained below.
- The Key Usage value shall be checked.
- The structure of the public key, including the existence of the referenced key parameters, shall be checked and the public key shall be extracted.
- All other fields of the certificate may be ignored.

If certificate chaining is not used, only CERT.OCE.ECKA is submitted and:

- CERT.OCE.ECKA shall be verified using the referenced PK.CA-KLOC.ECDSA (see P1 and P2 parameters).
- If no allowlist is defined for the referenced PK.CA-KLOC.ECDSA (see section 7.9), the CSN of CERT.OCE.ECKA shall not be checked. Otherwise, if this CSN is not referenced in the allowlist, then the certificate shall be rejected.

When certificate chaining and intermediate Key Authority(s) are used:

- The payload is composed of a chain of certificates starting from the first (or only) CERT.KA-KLOC.ECDSA and ending with CERT.OCE.ECKA, and this command shall be used to submit one certificate at a time (see usage of P1 and P2 parameters).
- The first (or only) CERT.KA-KLOC.ECDSA shall be verified using the referenced PK.CA-KLOC.ECDSA. Each subsequent certificate in the chain shall be verified using the public key extracted from the certificate immediately preceding it. In particular, the reference to PK.CA-KLOC.ECDSA given by the P1 and P2 parameters is only relevant for the first (or only) CERT.KA-KLOC.ECDSA, and shall be ignored for subsequent certificates.
- If no allowlist is defined for the referenced PK.CA-KLOC.ECDSA (see section 7.9), CSNs found in the certificate chain shall not be checked. Otherwise, if the CSN of the first (or only) CERT.KA-KLOC.ECDSA is not referenced in the allowlist, then the certificate (and the entire certificate chain) shall be rejected. Other CSNs found in the certificate chain shall not be checked (i.e. other certificates following in the chain are not required to be referenced in this allowlist in order to be accepted).
- The allowlist only contains CSNs of intermediate Key Authorities directly signed by the CA. Thus all certificates issued by any allowlisted intermediate Key Authority will be accepted.

If these verifications are successful, the SD shall extract PK.OCE.ECKA, the Subject Identifier (tag '5F20' for GP Certificates or 'id-sd-subject-id' extension for X.509 Certificates) and tag 'BF20' (if present) from CERT.OCE.ECKA and store them temporarily for use with the next MUTUAL AUTHENTICATE command. The following rules apply:

- If bit 3 of the “i” option is set (see section 4.2) and if tag 'BF20' was not present in the certificate, PK.OCE.ECKA shall instead be stored persistently for use with all subsequent MUTUAL AUTHENTICATE commands. Same for the Subject Identifier.
  - Only a single PK.OCE.ECKA shall be stored persistently per SD, hence storing a PK.OCE.ECKA persistently shall replace a PK.OCE.ECKA (if any) that had previously been persistently stored. Same for the Subject Identifier.
- Otherwise, the PK.OCE.ECKA is to be temporarily stored only. In this case, it shall take precedence over a persistently stored PK.OCE.ECKA (if any) on the next MUTUAL AUTHENTICATE command. Same for the Subject Identifier.

Note 1: *The Subject ID shall subsequently be used to determine if the established security level is AUTHENTICATED or ANY_AUTHENTICATED.*

Note 2: *An update of a PK.CA-KLOC.ECDSA does not have any effect on a stored PK.OCE.ECKA. If the OCE’s static key pair is (also) no longer trusted, the PK.OCE.ECKA stored in the SD has to be updated by a separate PERFORM SECURITY OPERATION command.*

7.5.2 Command Message

The PERFORM SECURITY OPERATION command message shall be coded according to the following table.

Table 7-8: PERFORM SECURITY OPERATION Command Message

Code	Value	Meaning
CLA	'80' - '87', 'C0' - 'CF', or 'E0' - 'EF'	See [GPCS] section 11.1.4.
INS	'2A'	PERFORM SECURITY OPERATION
P1	'xx'	Key Version Number and command chaining information
P2	'xx'	Key Identifier and certificate chaining information
Lc	'xx'	Length of data field
Data	'xx xx...'	Certificate
Le	'00'	

Note: *The command can also be sent in a secure channel session (see range for CLA), which will modify the data structure (e.g. adding MACs, etc.). A subsequent MUTUAL AUTHENTICATE command will terminate this session and initiate a new secure channel session.*

7.5.2.1 Reference Control Parameter P1

Reference control parameter P1 references the Key Version Number of the PK.CA-KLOC.ECDSA, which is used to verify the certificate's signature. It is coded on bits 1 to 7. Bit 8 is used for the command chaining information.

The command chaining information indicates whether the data field of the command is an intermediate or the last block of the currently submitted certificate.

Table 7-9: Values of Reference Control Parameter P1

b8	b7	b6	b5	b4	b3	b2	b1	Description
X								Command chaining information: 1: More command(s) 0: Last command
	X	X	X	X	X	X	X	Key version number of the PK.CA-KLOC.ECDSA

If the certificate doesn't exceed 255 bytes, the command chaining information (b8) shall be set to 0 (Last command). Otherwise, bit 8 shall be set to 1 (More command(s)) until the last block of the certificate.

7.5.2.2 Reference Control Parameter P2

Reference control parameter P2 references the Key Identifier of the PK.CA-KLOC.ECDSA, which is used to verify the certificate's signature. It is coded on bits 1 to 7. Bit 8 is used for the certificate chaining information.

The certificate chaining information indicates whether the submitted certificate (in the data field of the command) is an intermediate or the last certificate of the chain i.e. intermediate CERT.KA-KLOC.ECDSA or final CERT.OCE.ECKA.

Table 7-10: Values of Reference Control Parameter P2

b8	b7	b6	b5	b4	b3	b2	b1	Description
X								Certificate chaining information: 1: More certificate(s) 0: Last certificate
	X	X	X	X	X	X	X	Key identifier of the PK.CA-KLOC.ECDSA

Bit 8 shall be set to 0 (Last certificate) if the submitted certificate is CERT.OCE.ECKA, i.e. the last certificate. Otherwise, for intermediate certificates, bit 8 shall be set to 1 (More certificate(s) expected).

If bit 8 is set to 1 and SCP options (see Table 4-1) indicate that certificate chains are not supported, then a response of '6A86' shall be returned.

7.5.2.3 Example of Using P1P2

Given a certificate chain (CERT1…CERTn and CERT.OCE.ECKA) and certificates exceeding 255 bytes, the scenario is as follows:

Table 7-11: Example of Using P1P2

	Cmd N°	P1.b8	P2.b8
CERT1	0	More command(s)	More certificate(s)
	…		
	N-1		
	N	Last command	
…			
CERTn	0	More command(s)	More certificate(s)
	…		
	N-1		
	N	Last command	
CERT.OCE.ECKA	0	More command(s)	Last certificate
	…		
	N-1		
	N	Last command	

Note: *In our example, CERT1 is verified using the preloaded PK.CA-KLOC.ECDSA and CERT.OCE.ECKA is verified using the extracted public key (PK.CERTn) from the CERTn.*

7.5.2.4 Data Field Sent in the Command Message

The data field of the command message shall contain part or all of the data described in the following table. Such data may be split across several consecutive PERFORM SECURITY OPERATION commands as described by the P1 and P2 parameters.

Table 7-12: PERFORM SECURITY OPERATION Command Data

Certificates List	MOC
Certificate (CERT.KA-KLOC.ECDSA)	O
...	O
Certificate (CERT.OCE.ECKA)	M

Certificate formats are defined in section 6. All certificates in this list shall be encoded in the same format.

7.5.3 Response Message

7.5.3.1 Data Field Returned in the Response Message

The data field of the response message shall not be present.

7.5.3.2 Processing State Returned in the Response Message

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

This command may either return a general error condition as listed in [GPCS] section 11.1.3 or one of the following error conditions.

Table 7-13: PERFORM SECURITY OPERATION Error Conditions

SW1	SW2	Meaning
'66'	'00'	Verification of the certificate failed
'66'	'40'	Certificate not in allowlist
'6A'	'80'	Incorrect values in command data
'6A'	'81'	'BF20' authorization mechanism not supported
'6A'	'88'	Referenced PK.CA-KLOC.ECDSA not found
'6A'	'86'	Certificate chains not supported
'6A'	'87'	Certificate format not supported

7.5.4 Processing Rules

When the Security Level established for an SCP11a or SCP11c secure channel session is ANY_AUTHENTICATED, tag 'BF20' allows to selectively authorize the actions that may be performed during the session (i.e. superseding the rules defined in [GPCS] Table 11-2). For details, see Annex B. This tag may be present for a key agreement certificate (i.e. value '00 80' for tag '95') which is to be used to open an SCP11a or SCP11c session. Finally, it shall be absent in a signature verification certificate (i.e. value '82' for tag '95').

If a Security Domain does not support the 'BF20' authorization mechanism (as indicated by its SCP11 “i” option) and tag 'BF20' is present, then the command shall fail with an error of '6A81'.

7.6 MUTUAL AUTHENTICATE Command

7.6.1 Definition and Scope

The MUTUAL AUTHENTICATE command is used to send the ephemeral public key of the OCE to the SD, to trigger the key establishment, to provide card authentication information to the OCE, and to determine the level of security required for all subsequent commands.

The MUTUAL AUTHENTICATE command terminates any ongoing secure channel session (whichever secure channel protocol is currently used) on the same logical channel and if the command is successful, initiates a new secure channel session.

If no PK.OCE.ECKA was provided earlier to the SD using a PERFORM SECURITY OPERATION command or if an allowlist was updated since PK.OCE.ECKA was provided (see section 7.9), the MUTUAL AUTHENTICATE command shall fail with error condition “Conditions of use not satisfied”.

If the PK.OCE.ECKA was not provided immediately before the MUTUAL AUTHENTICATE command, the OCE should check that the SD used the correct PK.OCE.ECKA by verifying the receipt generated by the SD.

7.6.2 Command Message

The MUTUAL AUTHENTICATE command message is coded according to the following table.

Table 7-14: MUTUAL AUTHENTICATE Command Message

Code	Value	Meaning
CLA	'80' - '83' or 'C0' - 'CF'	See [GPCS] section 11.1.4.
INS	'82'	MUTUAL AUTHENTICATE
P1	'xx'	Key Version Number
P2	'xx'	Key Identifier
Lc	'xx'	Length of data field
Data	'xx xx…'	Data for key establishment
Le	'00'	

Note: *INS for MUTUAL AUTHENTICATE is the same as for EXTERNAL AUTHENTICATE used in SCP02 and SCP03. However, the P2 value, which is set to '00' for SCP02 and SCP03, is always different for SCP11.*

7.6.2.1 Reference Control Parameter P1

Reference control parameter P1 references the Key Version Number of the SK.SD.ECKA. It is coded on bits 1 to 7. Bit 8 is RFU and set to zero.

7.6.2.2 Reference Control Parameter P2

Reference control parameter P2 references the Key Identifier of the SK.SD.ECKA. It is coded on bits 1 to 7. Bit 8 is RFU and set to zero.

7.6.2.3 Data Field Sent in the Command Message

The data field of the command message shall be coded according to the following table.

Table 7-15: MUTUAL AUTHENTICATE Data Field

Tag	Length	Value Description			MOC
'A6'	Var	Control Reference Template (Key Agreement)			M
		Tag	**Length**	**Value Description**	**MOC**
		'90'	2	SCP identifier and parameters (see section 7.1.1)	M
		'95'	1	Key Usage Qualifier • '34' (secure messaging with C-MAC and R-MAC) • '3C' (secure messaging with C-MAC, C-ENC, R-MAC, and R-ENC) • '1C' (secure messaging with C-MAC and C-ENC, only for SCP11c) • '74' (secure messaging with C-MAC, C-ENC, and R-MAC, only for SCP11c) (See [GPCS] Table 11-17)	M
		'80'	1	Key Type according to [GPCS] Table 11-16 • '88' (AES)	M
		'81'	1	Key Length (in bytes)	M
		'84'	1-n	HostID (shall only be present if SCP parameter b3 is set)	C
'5F49'	Var	ePK.OCE.ECKA			M

The SD shall verify the values provided for SCP identifier, SCP parameters, key usage qualifier, and key type.

The key usage qualifier (tag '95') shall be used to specify the Security Level of the SCP11 session. The usage of values '1C' and '74' is only allowed for SCP11c.

If mandated by the security policy, key length shall be checked according to the recommendations defined in section 5.2.

If bit 3 of the SCP parameters is set (“Include Host and Card ID in key derivation process”) and tag '84' (Host ID) is not present within tag 'A6', then an error shall be returned. Similarly, if bit 3 is not set and tag '84' (Host ID) is present within tag 'A6', then an error shall be returned.

In the case of SCP11a:

- If the SD does not support tag 'BF20' for SCP11a, it shall check that the CERT.OCE.ECKA provided by the previous PERFORM SECURITY OPERATION command did not contain tag 'BF20'.
- The SD shall generate an ephemeral key pair eSK.SD.ECKA and ePK.SD.ECKA.
- The SD shall use PK.OCE.ECKA and SK.SD.ECKA to generate the shared secret ShSss according to section 3.1.1.
- The SD shall use ePK.OCE.ECKA and eSK.SD.ECKA to generate the shared secret ShSee according to section 3.1.1.

- The SD shall concatenate ShSee and ShSss to form the shared secret ShS which constitutes the input for the Key Derivation process.

In the case of SCP11c:

- If the SD does not support tag 'BF20' for SCP11c, it shall check that the CERT.OCE.ECKA provided by the previous PERFORM SECURITY OPERATION command did not contain tag 'BF20'.
- The SD shall use PK.OCE.ECKA and SK.SD.ECKA to generate the shared secret ShSss according to section 3.1.1.
- The SD shall use ePK.OCE.ECKA and SK.SD.ECKA to generate the shared secret ShSes according to section 3.1.1.
- The SD shall concatenate ShSes and ShSss to form the shared secret ShS which constitutes the input for the Key Derivation process.

The concatenation of the following values shall be used for *SharedInfo* as input for the Key Derivation process:

- Key usage qualifier (1 byte)
- Key type (1 byte)
- Key length (1 byte)
- If Host and Card ID are requested:
  - In the case of SCP11a and SCP11b: HostID-LV, SIN-LV, and SDIN-LV
  - In the case of SCP11c: HostID-LV and Card Group ID-LV

  Note: *The presence of unique host (Off-Card Entity) and card identifiers is required in [NIST 800-56A].*

HostID-LV is the length and the value field of the HostID given in the command data.

SIN-LV is the length and the value field of the Security Domain Provider Identification Number of the SD (see [GPCS]).

SDIN-LV is the length and the value field of the Security Domain Image Number of the SD (see [GPCS]).

Card Group ID-LV is:

- In the case of GP Certificate: the length and the value field of tag '5F20' (subject identifier) in CERT.SD.ECKA.
- In the case of X.509 Certificate:
  - If present, the length and the value field of 'id-sd-subject-id' extension
  - Else the length and the value field of Distinguished Name

Note: *The OCE may need to obtain CERT.SD.ECKA before creating the SCP11c script to obtain PK.SD.ECKA.*

SHA-256 shall be used for the key derivation to calculate *KeyData* of sufficient length, which is then assigned to keys as defined below.

Note: *SHA-256 is considered strong enough even for AES-256 keys, and the output size aligns nicely with most key lengths.*

In addition to the session keys, a receipt key is used to calculate the receipt to be included in the response to the MUTUAL AUTHENTICATE command. The type and length of the receipt key is the same as for the session keys.

The *KeyData* generated as defined in section 3.1.2 shall be assigned to the keys as follows (L is the key length):

Table 7-16: *KeyData* Assignment

KeyData	Key
1 to L	Receipt key
L+1 to 2L	S-ENC
2L+1 to 3L	S-MAC
3L+1 to 4L	S-RMAC
4L+1 to 5L	S-DEK (if no Key-DEK is present)

Finally, the SD shall generate a receipt (using the receipt key and the MAC algorithm used in the secure channel) by calculating a MAC across the data described in Table 7-17. The receipt key shall be deleted after calculating the receipt.

Table 7-17: Input Data for Receipt Calculation

Tag	Length	Data Element	MOC
'A6'	Variable	CRT TLV with all sub TLVs as provided in the MUTUAL AUTHENTICATE command	M
'5F49'	Variable	ePK.OCE.ECKA	M
'5F49'	Variable	SCP11a: ePK.SD.ECKA SCP11c: PK.SD.ECKA	M

7.6.3 Response Message

7.6.3.1 Data Field Returned in the Response Message

The data field of the response message shall contain the data objects listed in the following table.

Table 7-18: MUTUAL AUTHENTICATE Response Data

Tag	Length	Value Description	MOC
'5F49'	Variable	SCP11a: ePK.SD.ECKA SCP11c: PK.SD.ECKA	M
'86'	16	Receipt	M

7.6.3.2 Processing State Returned in the Response Message

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

This command may either return a general error condition as listed in [GPCS] section 11.1.3 or one of the following error conditions.

Table 7-19: MUTUAL AUTHENTICATE Error Conditions

SW1	SW2	Meaning
'6A'	'80'	Incorrect values in command data
'6A'	'88'	One of the following referenced data elements is not found: SK.SD.ECKA / PK.OCE.ECKA / SIN / SDIN
'69'	'85'	Attempt to initiate an SCP11c session but one SCP11c session is already ongoing on another logical channel and the implementation does not support multiple SCP11c sessions

7.7 INTERNAL AUTHENTICATE Command

7.7.1 Definition and Scope

The INTERNAL AUTHENTICATE command is used to trigger the key establishment, to provide card authentication information to the OCE, and to determine the level of security required for all subsequent commands.

The INTERNAL AUTHENTICATE command terminates an ongoing secure channel session (whichever secure channel protocol is currently used) and if the command is successful, initiates a new secure channel session.

7.7.2 Command Message

The INTERNAL AUTHENTICATE command message is coded according to the following table.

Table 7-20: INTERNAL AUTHENTICATE Command Message

Code	Value	Meaning
CLA	'80' - '83' or 'C0' - 'CF'	See [GPCS] section 11.1.4.
INS	'88'	INTERNAL AUTHENTICATE
P1	'xx'	Key Version Number
P2	'xx'	Key Identifier
Lc	'xx'	Length of data field
Data	'xx xx…'	Data for key establishment
Le	'00'	

7.7.2.1 Reference Control Parameter P1

Reference control parameter P1 references the Key Version Number of the SK.SD.ECKA. It is coded on bits 1 to 7. Bit 8 is RFU and set to zero.

7.7.2.2 Reference Control Parameter P2

Reference control parameter P2 references the Key Identifier of the SK.SD.ECKA. It is coded on bits 1 to 7. Bit 8 is RFU and set to zero.

7.7.2.3 Data Field Sent in the Command Message

The data field of the command message shall be coded according to the following table.

Table 7-21: INTERNAL AUTHENTICATE Data Field

Tag	Length	Value Description			MOC
'A6'	Var	Control Reference Template (Key Agreement)			M
		Tag	**Length**	**Value Description**	**MOC**
		'90'	2	SCP identifier and parameters (see section 7.1.1)	M
		'95'	1	Key Usage Qualifier • '34' (secure messaging with C-MAC and R-MAC only) or • '3C' (secure messaging with C-MAC, R_MAC, and ENCRYPTION) (See [GPCS] Table 11-17)	M
		'80'	1	Key Type according to [GPCS] Table 11-16 • '88' (AES)	M
		'81'	1	Key Length (in bytes)	M
		'84'	1-n	HostID (shall only be present if SCP parameter b3 is set)	C
'5F49'	Var	ePK.OCE.ECKA			M

Processing shall be done as defined for MUTUAL AUTHENTICATE in section 7.6.2.3, with the following modifications:

- The SD shall generate an ephemeral key pair eSK.SD.ECKA and ePK.SD.ECKA.
- The SD shall use ePK.OCE.ECKA and SK.SD.ECKA to generate the shared secret ShSes according to section 3.1.1.
- The SD shall use ePK.OCE.ECKA and eSK.SD.ECKA to generate the shared secret ShSee according to section 3.1.1.
- The SD shall concatenate ShSee and ShSes to form the shared secret ShS which constitutes the input for the Key Derivation process.

7.7.3 Response Message

7.7.3.1 Data Field Returned in the Response Message

The data field of the response message shall contain the data objects listed in the following table.

Table 7-22: INTERNAL AUTHENTICATE Response Data

Tag	Length	Value Description	MOC
'5F49'	Variable	ePK.SD.ECKA	M
'86'	16	Receipt	M

7.7.3.2 Processing State Returned in the Response Message

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

This command may either return a general error condition as listed in [GPCS] section 11.1.3 or one of the following error conditions.

Table 7-23: INTERNAL AUTHENTICATE Error Conditions

SW1	SW2	Meaning
'6A'	'88'	One of the following referenced data elements is not found: SK.SD.ECKA / PK.OCE.ECKA / SIN / SDIN
'6A'	'80'	Incorrect values in command data

7.8 STORE DATA (Certificate Store) Command

The STORE DATA command is defined in [GPCS] section 11.11. BER-TLV format shall be used for the command data.

To store or replace an SCP11 certificate store, the data field of the command message shall contain two BER-TLVs as defined in the following table.

Table 7-24: Data Field of STORE DATA (Certificate Store) Command

Tag	Length	Value Description			MOC
'A6'	4	Control Reference Template (Key Agreement)			M
		Tag	**Length**	**Value Description**	**MOC**
		'83'	2	byte 1: Key Identifier byte 2: Key Version Number	M
'BF21'	Var	SCP11 certificate store (see section 7.2, Table 7-4)			M

An SCP11 certificate store shall contain certificates either in GP legacy format or in X.509 format, but not both.

If certification chaining and one or more Key Authorities are used, the SCP11 certificate store shall contain all the certificates in suitable verification order, i.e. starting from the one signed by CA-KLCC and ending with CERT.SD.ECKA. Except for the first certificate (that shall be verified using PK.CA-KLCC.ECDSA), each certificate shall be verified by the public key extracted from the certificate preceding it.

After successful execution of the command, an SCP11 certificate store is linked to the private key SK.SD.ECKA referenced by the CRT. This certificate store may contain a chain of certificates ending with CERT.SD.ECKA as shown in section 7.2, Table 7-4.

When the SK.SD.ECKA is deleted or replaced, the SCP11 certificate store shall be automatically deleted by the SD.

If the referenced SK.SD.ECKA does not exist in the SD, the command shall be rejected with error condition '6A88'.

7.9 STORE DATA (Allowlist) Command

The STORE DATA command is defined in [GPCS] section 11.11. BER-TLV format shall be used for the command data.

To store or replace an allowlist linked to a PK.CA-KLOC.ECDSA, the data field of the command message shall contain three BER-TLVs as defined in the following table.

Update of the allowlist shall only be accepted if the Security Level is AUTHENTICATED.

Note: *Security Level AUTHENTICATED cannot be achieved with SCP11b.*

Table 7-25: Data Field of STORE DATA (Allowlist) Command

Tag	Length	Value Description			MOC
'A6'	4	Control Reference Template (Key Agreement)			M
		Tag	**Length**	**Value Description**	**MOC**
		'83'	2	byte 1: Key Identifier byte 2: Key Version Number	M
'92'	2	Allowlist counter (2-byte positive integer up to 32767)			C
'70'	Var	Allowlist			M
		Tag	**Length**	**Value Description**	**MOC**
		'93'	Var.	Certificate Serial Number	O
		'93'	Var.	Certificate Serial Number	O
		…	…	…	…
		'93'	Var.	Certificate Serial Number	O

Note: *For an X.509 certificate, tag '93' (Certificate Serial Number) relates to the 'serialNumber' filed.*

After successful execution of the command, the allowlist of the command replaces any previously stored allowlist linked to the PK.CA-KLOC.ECDSA referenced by the CRT. To remove an allowlist, the allowlist TLV of the command shall have a length of zero.

When an allowlist is newly stored or replaced, a stored PK.OCE.ECKA shall be deleted by the SD.

When a PK.CA-KLOC.ECDSA is deleted, an allowlist linked to it shall also be deleted by the SD. On the other hand, the update of a PK.CA-KLOC.ECDSA shall have no impact on a linked allowlist.

If the SD supports SCP11c (according to its "i" parameter), then:

- The SD shall maintain an internal anti-replay allowlist counter for each PK.CA-KLOC.ECDSA. This counter shall never be reset, even if the allowlist is deleted (i.e. the same counter shall be used for any allowlist subsequently added for the same PK.CA-KLOC.ECDSA).
- Tag '92' shall be present in the command if it is received in an SCP11c session and shall not be present otherwise.

- If this command is received in an SCP11c session, then:
  - If any of the following conditions is true:
    - The allowlist counter (tag '92') is missing in the command, or
    - The value of the internal counter is higher than or equal to the allowlist counter sent in the command, or
    - The internal counter value has reached its maximum value (32767),

    ...then an error of '6A80' shall be returned.
  - If the update of the allowlist is successful, then the value of the internal counter shall be updated to the value of tag '92'. Notice that the allowlist and the internal counter shall be updated atomically in the same transaction.

7.10 STORE DATA (CA Identifier) Command

The STORE DATA command is defined in [GPCS] section 11.11. BER-TLV format shall be used for the command data.

To store or replace a CA-KLOC Identifier linked to a PK.CA-KLOC.ECDSA or a CA-KLCC Identifier linked to a SK.SD.ECKA, the data field of the command message shall contain a BER-TLV as defined in the following table.

Table 7-26: Data Field of STORE DATA (CA-KLOC Identifier) Command

Tag	Length	Value Description			MOC
'A6'	Var	Control Reference Template			M
		Tag	**Length**	**Value Description**	**MOC**
		'80'	1	0: CA-KLOC (default) or 1: CA-KLCC	O
		'42'	Var.	CA Identifier	M
		'83'	2	byte 1: Key Identifier byte 2: Key Version Number	M

Sub-tag '80' indicates whether the CA Identifier being stored relates to a CA-KLOC or CA-KLCC. If not present, the CA Identifier relates to a CA-KLOC by default.

After successful execution of the command, the CA-KLOC Identifier (resp. CA-KLCC Identifier) is linked to the PK.CA-KLOC.ECDSA (resp. SK.SD.ECKA) referenced by the specified KID and KVN.

When a PK.CA-KLOC.ECDSA is deleted or replaced, the associated CA-KLOC Identifier shall be automatically deleted by the SD. Similarly, when a SK.SD.ECKA is deleted or replaced, the associated CA-KLCC Identifier shall be automatically deleted by the SD.

If the referenced PK.CA-KLOC.ECDSA or SK.SD.ECKA does not exist in the SD, the command shall be rejected with error condition '6A88'.

Table 7-27: STORE DATA Error Conditions

SW1	SW2	Meaning
'6A'	'80'	Incorrect values in command data
'6A'	'88'	The referenced PK.CA-KLOC.ECDSA does not exist in the SD

7.11 DELETE

The DELETE command is defined in [GPCS] section 11.2. The following sections define new usages for this command.

7.11.1 SD Self-Deletion

This specification defines a new mechanism which allows a Security Domain to delete itself and all dependent packages and instances. At the end of this process, all resources associated with this SD will be deleted.

In the context of this specification, this mechanism is primarily intended for use within a SCP11c session, hence it shall be supported by the implementation when the SCP11c protocol is supported and it shall be possible to use it within a SCP11c session. An implementation may support this mechanism to be used within other types of Secure Channel sessions (not limited to SCP11).

A DELETE command shall be recognized as a self-deletion command if the following conditions are met:

- The AID to be deleted (contents of tag '4F') is identical to the AID of the SD instance that receives the command and
- The reference control parameter P2 is set accordingly (P2.b6=1).

The following table shows the values for the reference control parameter P2 used to indicate self-deletion:

Table 7-28: Values for Reference Control Parameter P2

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
0	-	-	-	-	-	-	-	
1	-	-	-	-	-	-	-	Delete object and related object
0	1	-	-	-	-	-	-	Delete a root Security Domain and all associated Applications
0	0	1	-	-	-	-	-	Self-Delete Security Domain
-	-	-	X	X	X	X	X	RFU

When a self-deletion command is received, the following process shall be executed:

- If the SD or any of its dependent SDs or Applications is selected on another logical channel, the command shall be rejected.
- The SD shall:
  - Check that the current Security Level is AUTHENTICATED (or higher).
- The SD shall request the OPEN to apply Cumulative Deletion to itself (as defined in [Amd C] but not limited to a root Security Domain). Doing so, the OPEN shall:
  - Check that the card Life Cycle State is neither CARD LOCKED nor TERMINATED.
  - Deselect the SD (note that this has the effect of closing the secure channel).
  - Perform Cumulative Deletion of the SD.
  - If Cumulative Deletion was successful, return a status word of '9000'. Otherwise, return an error status word.

7.11.2 SD Cumulative Deletion

The provisions of [Amd C] section 10 of (Cumulative DELETE) shall apply if the following conditions are met:

- The Security Domain that processes the DELETE command has the Global Delete privilege.
- The AID to be deleted refers to a Security Domain.
- The reference control parameter P2 of the DELETE command is set to “Delete a root Security Domain and all associated Applications” (see [Amd C]).

  NOTE: Although the P2 option says "root Security Domain", cumulative deletion would apply to the referenced SD at any level.

8 CARD CONTENT TRANSACTION

This section describes a mechanism allowing to manage the creation of new card contents within an atomic transaction, which may be useful to prevent incomplete setup of card contents in case of interruption. Within such a transaction, called Card Content Transaction (CCT), new Executable Load Files may be loaded and new Applications and/or Security Domains may be installed and personalized. If the transaction is interrupted, all the entities loaded or installed during the transaction shall be deleted. Notice that this transaction mechanism is not expected to log and undo updates performed (during the transaction) to entities existing/created prior to the start of the transaction.

In the context of this specification, the CCT mechanism is primarily intended for use within a SCP11c session, hence it shall be supported by the implementation when the SCP11c protocol is supported. The PERFORM TRANSACTION OPERATION command described in section 8.2 shall be supported within a SCP11c session. An implementation may support this command to be used within other types of Secure Channel sessions (not limited to SCP11).

8.1 Detection and Removal of Pending Card Contents

The Card Content Transaction mechanism works as follows:

- The PERFORM TRANSACTION COMMAND (see section 8.2) shall be used to begin or commit a transaction. It shall be processed within a Secure Channel session.
- The transaction is bound to the Secure Channel session, logical channel and I/O interface on which it is started. New contents shall be marked (see details below) when created by a command processed within the Secure Channel session bound to the transaction. New contents created by commands processed outside that Secure Channel session (e.g. on other logical channels) shall not be marked as part of the transaction.
- If a new ELF is loaded or a new Applet (or Security Domain) instance is installed during an ongoing transaction, it shall be marked internally as “pending completion”.
- All “pending completion” marks shall be cleared when a PERFORM TRANSACTION COMMAND [commit] command is received.
- If any of the following events occurs during an ongoing transaction, all objects having a “pending completion” mark shall be deleted as soon as possible (and before processing the next APDU command):
  - The Secure Channel session bound to the transaction is closed. See [GPCS] for Secure Channel Termination.
  - The logical channel bound to the transaction is closed.
  - The card is powered down or the I/O interface (contact, contactless) bound to the transaction is reset.
  - An attempt to open a new Secure Channel session is made on the same logical channel.

Note 1: *This transaction mechanism only provides limited guarantees that the card will be in the same state as before in case of rollback. For example, any modification performed on some data that does not belong to an Applet instance created during the transaction may persist after the rollback.*

Note 2: *A command failing for a reason other than a secure messaging error is not defined in this mechanism as an event that shall trigger the abortion of the transaction. Moreover, the SD processing the command also doesn’t have any clue as to whether it should abort the transaction in such cases. If aborting the transaction is the desired course of action, the Off-Card Entity providing the command may trigger the abortion by closing the Secure Channel session, which may be simply achieved by, for example, selecting another Application or re-selecting the SD on the same logical channel or closing the logical channel on which the latter is selected.*

8.2 PERFORM TRANSACTION OPERATION Command

8.2.1 Definition and Scope

The PERFORM TRANSACTION OPERATION command shall be used to begin or commit a Card Content Transaction. This command shall be processed within a Secure Channel session.

8.2.2 Command Message

The PERFORM TRANSACTION OPERATION command message is coded according to the following table.

Table 8-1: PERFORM TRANSACTION OPERATION Command Message

Code	Value	Meaning
CLA	'84' - '87' or 'E0' - 'EF'	See [GPCS] section 11.1.4.
INS	'12'	PERFORM TRANSACTION OPERATION
P1	'00'	
P2	'80'	BEGIN TRANSACTION
	'81'	COMMIT TRANSACTION
Lc		Not present
Data		Not present
Le		Not present

- If a Security Domain receives a BEGIN TRANSACTION command, the Security Domain shall start marking newly installed objects as “pending completion” (see section 8.1).
  - If two consecutive BEGIN TRANSACTION commands are received with no COMMIT TRANSACTION command in between, the Secure Channel session shall be terminated and all objects with a “pending completion” mark shall be deleted.
- If a Security Domain receives a COMMIT TRANSACTION command, the Security Domain shall remove the “pending completion” mark from all objects created during the Card Content Transaction.

8.2.3 Response Message

The data field of the response message is empty.

This command may either return a general error condition, as listed in [GPCS] section 11.1.3, or one of the following error conditions:

Table 8-2: PERFORM TRANSACTION OPERATION Error Conditions

SW1	SW2	Meaning
'69'	'85'	A COMMIT TRANSACTION was received without a prior BEGIN TRANSACTION
'6A'	'84'	Not enough resources to begin a new transaction

Annex A OCE AUTHENTICATION FOR SCP11b

SCP11b provides authentication of the card to the OCE only. Mechanisms for authentication of the OCE to the card are out of scope of the secure channel protocol and have to be provided by applications that are using SCP11b.

This annex provides an example for a mechanism based on PIN verification which could be used for this purpose.

A.1 OCE Providing PIN Verification

A weak authentication mechanism can be provided by the OCE by sending a PIN code to the SD, which was entered by the user at the OCE's user interface. Strictly speaking, this authenticates the user. However, if the user is instructed to enter the PIN only on the user interface of the device hosting the OCE's endpoint of the secure channel, this indirectly also authenticates the OCE.

This approach may be used in certain cases where the OCE is (a Trusted Application) in a Trusted Execution Environment, providing a Trusted User Interface (see the GlobalPlatform TEE Trusted User Interface Low-level API, [TEE TUI Low]).

The detailed interaction between the card and the device is out of scope of this specification.

> ***Note: Assuming PIN verification would be implemented by the SD, whether successful verification should have an impact on the actual security level of the SCP11b session (i.e. AUTHENTICATED or ANY_AUTHENTICATED) is out of scope and policy dependent.***

This annex provides an example of the command used by the application in such a scenario once SCP11b is established: VERIFY PIN. The VERIFY PIN command message in the secure channel is coded according to the following table.

Table A-1: VERIFY PIN Command Message

Code	Value	Meaning
CLA	'84' – '87', or 'E0' – 'EF'	Please refer to [GPCS] section 11.1.4
INS	'20'	VERIFY PIN
P1	'00'	Reference control parameter P1: Normal operation
P2	'00'	Reference control parameter P2: No information given
Lc	'xx'	Length of data field
Data	'xx xx…'	PIN value
Le		Not present

A.1.1 Data Field Sent in the Command Message

The data field contains the UTF-8 encoded PIN entered by the user.

A.1.2 Processing State Returned in the Response Message

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

This command may either return a general error condition as listed in [GPCS] section 11.1.3 or one of the following error conditions.

Table A-2: VERIFY PIN Error Conditions

SW1	SW2	Meaning
'63'	'CX'	Authentication failed, X retries allowed
'69'	'83'	Authentication failed, PIN blocked

Annex B USAGE OF SCP11a AND SCP11c

This document introduces some changes to the behavior of the card if SCP11a and SCP11c has been set up successfully. These changes are described in this annex.

B.1 Authorization Rules for SCP11a and SCP11c

In addition to usual APDU command processing rules described in [GPCS], APDU command processing authorization rules may be specified in tag 'BF20' of the OCE certificate (CERT.OCE.ECKA) used to establish the SCP11a or SCP11c session. Such authorization rules shall only be checked and enforced under the following conditions:

- The security level indicates ANY_AUTHENTICATED; **and**
- The SD processing the APDU command has the Authorized Management privilege; **and**
  - The APDU command does not require a Delegated Management Token (e.g. STORE DATA); **or**
  - The APDU command requires a Delegated Management Token (see [GPCS] section 9.1.3.2, e.g. INSTALL [for install]…) but this token is missing in the command. In this case, the authorization rules shall be checked as a replacement for token verification.

In particular, such authorization rules shall be ignored in any of the following conditions:

- The security level indicates AUTHENTICATED. In this case, further processing of the command is allowed.
- The SD processing the APDU command does not have either the Authorized Management or the Delegated Management privilege, but processing the command (e.g. INSTALL [for install]) requires the SD to have one of these privileges. In this case, the command shall be rejected.
- The SD processing the APDU command has the Delegated Management privilege and the APDU command requires a Delegated Management Token. In this case, further processing of the command shall be allowed based on the presence and verification of the token.
- The SD processing the APDU command has the Authorized Management privilege, the security level indicates ANY_AUTHENTICATED, and the APDU command contains a Delegated Management Token. In this case, further processing of the command shall be allowed based on the verification of the token.

See also section 4.6 for a description of commands that shall always be rejected within an SCP11c session.

Table B-1 describes how authorization rules may be specified in the OCE certificate as part of tag 'BF20', as well as the default behavior that shall apply when some authorization rules are not specified.

Table B-1: Contents of Tag 'BF20' in OCE Certificate

Tag	Length	Value Description			MOC
'E3'	Var	GlobalPlatform Registry related data			
		Tag	**Length**	**Value Description**	**MOC**
		'8A'	5-16	*Deprecated (see Tag '90' and Tag '91')* AID prefix bytes used to control which Application may be installed, and/or which ELF may be loaded. Only ELF and Application AID values matching these prefix bytes (i.e. partial AID matching) may be accepted in the following commands: • INSTALL [for load] • INSTALL [for install] • INSTALL [for make selectable] • INSTALL [for install & make selectable] • INSTALL [for extradition] For this command, only the Application AID shall be checked, not the one of the target SD. • INSTALL [for registry update] For this command, only the Application AID shall be checked, not the one of the target SD. • INSTALL [for personalization] For this command, only the Application AID shall be checked. If this tag is not present, the INSTALL command is not allowed. Several occurrences of this tag may be present to provide several patterns.	C
		'C5'	3	Privileges that may be assigned to an Application created through this secure channel. Only the specified privileges may be assigned in the following commands: • INSTALL [for install] • INSTALL [for make selectable] • INSTALL [for install & make selectable] Notice however that a Security Domain may not grant more privileges (to another Security Domain) than it has itself.	O

<table>
<tr><th>Tag</th><th>Length</th><th colspan="3">Value Description</th><th>MOC</th></tr>
<tr><td></td><td></td><td>'90'</td><td>5-16</td><td>Application AID prefix bytes used to control which Application may be installed.
Only Application AID values matching these prefix bytes (i.e. partial AID matching) may be accepted in the following commands:
• INSTALL [for install]
• INSTALL [for make selectable]
• INSTALL [for install & make selectable]
• INSTALL [for extradition]
For this command, only the Application AID shall be checked, not the one of the target SD.
• INSTALL [for registry update]
For this command, only the Application AID shall be checked, not the one of the target SD.
• INSTALL [for personalization]
For this command, only the Application AID shall be checked.
If this tag is not present, the INSTALL command is not allowed. Several occurrences of this tag may be present to provide several patterns. See tag '91' for INSTALL [for load] command</td><td>C</td></tr>
<tr><td></td><td></td><td>'91'</td><td>5-16</td><td>ELF AID prefix bytes used to control which Application may be loaded.
Only ELF matching these prefix bytes (i.e. partial AID matching) may be accepted in the following commands:
• INSTALL [for load]
• INSTALL [for install]
• INSTALL [for install & make selectable]
If this tag is not present, the INSTALL [for load] command is not allowed. Several occurrences of this tag may be present to provide several patterns.</td><td>C</td></tr>
<tr><td>'EF'</td><td>Var</td><td colspan="4">System specific parameters
Tag | Length | Value Description | MOC
'82' | 2 or 4 | Maximum cumulative granted volatile memory that may be defined for a Security Domain created through this secure channel.
If this tag is present, then:
• Any attempt to allocate a volatile CGM exceeding this value shall fail.
• Any attempt to install a new Security Domain shall include the specification of a volatile CGM value, unless such a value has already been defined for a Security Domain higher in the hierarchy of the new Security Domain. | O</td></tr>
</table>

Tag	Length	Value Description			MOC
		'83'	2 or 4	Maximum cumulative granted non-volatile memory that may be defined for a Security Domain created through this secure channel. If this tag is present, then: • Any attempt to allocate a non-volatile CGM exceeding this value shall fail. • Any attempt to install a new Security Domain shall include the specification of a non-volatile CGM value, unless such a value has already been defined for a Security Domain higher in the hierarchy of the new Security Domain.	O
		'8B'	Var	List of allowed DGI numbers in STORE DATA. The list is formatted as a sequence of the DGI tags. If this tag is not present, the DGI variant of STORE DATA is not allowed. Notice however that this rule only applies to a STORE DATA command targeting a Security Domain. No restrictions apply to a STORE DATA command received by a regular Application (directly or forwarded by its associated Security Domain).	O
		'8C'	Var	List of allowed TLV tags in STORE DATA. The list is formatted as a sequence of the BER-TLV tags. If this tag is not present, the TLV variant of STORE DATA is not allowed. Notice however that this rule only applies to a STORE DATA command targeting a Security Domain. No restrictions apply to a STORE DATA command received by a regular Application (directly or forwarded by its associated Security Domain).	O
		'8E'	5-16	AID prefix bytes used to control which Application or ELF may be deleted. Only Application and ELF AID values matching these prefix bytes (i.e. partial AID matching) may be accepted. If this tag is not present, the DELETE command is not allowed. Several occurrences of this tag may be present to provide several patterns. The cumulative DELETE command (P2.b7=1) as defined in Amendment C: Contactless Services ([Amd C]) shall be supported if the SD processing the SCP11c session has the Global Delete privilege.	O

Tag	Length	Value Description			MOC
		'8F'	5-16	AID prefix bytes used to control which Executable Load File may be upgraded. Only ELF AID values matching these prefix bytes (i.e. partial AID matching) may be accepted. If this tag is not present, the MANAGE ELF UPGRADE command is not allowed. Several occurrences of this tag may be present to provide several patterns.	O

Security Domains shall ignore unknown tags within tags 'E3' and 'EF' to allow for future extensibility.

> ***Note:*** *If SCP11c is only used to protect scenario #1 of [Amd A], the contents of tag '8B' should be '00 DE 00 A6' to authorize storing an AP certificate and trigger key generation. If SCP11c is only used to protect scenario #3 of [Amd A], the contents of tag '8B' should be '7F 49 00 A6' to authorize the intended key generation.*

If allowed, the commands shall be executed as described in [GPCS].

If the script attempts to perform an operation that is not allowed, the command triggering this operation shall be rejected with a status word of '6985', the SCP11a or SCP11c session shall be terminated.

Table B-2 summarizes, for each Security Domain APDU command, which sub-tags are required (within tag 'BF20') when the processing of APDU command is subject to authorization (as described at the beginning of this section):

Table B-2: Required Contents within Tag 'BF20' per APDU Command

Command	SCP11a	SCP11c
DELETE [AID]	Tag '8E' is required.	
DELETE [key(s)]	N/A (Always allowed)	N/A (Never allowed)
Cumulative DELETE	Tag '8E' is required.	
GET DATA	N/A (Always allowed)	
GET STATUS	N/A (Always allowed)	N/A (Never allowed)
INSTALL	The following requirements apply to specific INSTALL commands:	
[for registry update]	Tag '8A' (deprecated) or '90' is required. Tag 'C5' is required if privileges are updated.	
[for personalization]	Tag '8A' (deprecated) or '90' is required.	
[for extradition]	Tag '8A' (deprecated) or '90' is required. Tag 'C5' is required.	
[for install]		
[for make selectable]		
[for install & make selectable]		
[for load]	Tag '8A' (deprecated) or '91' is required.	
LOAD	N/A (Always allowed) NOTE: must be preceded by a valid INSTALL [for load] command.	
MANAGE ELF UPGRADE	Tag '8F' is required.	
PERFORM TRANSACTION OPERATION	N/A (Always allowed)	
PUT KEY	N/A (Always allowed)	N/A (Never allowed)
SET STATUS	N/A (Always allowed)	N/A (Never allowed)
SELECT	N/A (Always allowed) NOTE: will terminate the SCP11 session.	
STORE DATA	Tag '8B' or '8C' is required.	

Annex C VOID

Annex D ID ALLOCATION (INFORMATIVE)

The value of the GlobalPlatform root OID is:

joint-iso-itu-t(2) country-USA(840) gp(114283)

To this project, a node was allocated:

card-management(100) modules(0) scp11(1)

All OIDs allocated in this version of this specification belong to the scp11 node:

scp11(1) – root for the SCP11 project

- spec-version(1) – root for identifying the ASN.1 module of the different versions
  - version-one(1) – ASN.1 module of version 1.X
  - version-two(2) – ASN.1 module of version 2.X
  - version-three(3) – ASN.1 module of version 3.X
  - … – future ASN.1 modules SHOULD use additional sub-nodes here
- cert-objects(2) – root for nodes identifying objects and roles used in certificates
  - id-scp11Ext(0) – root for certificate extensions
    - id-disc-data(0)
    - id-scp11-auth(1)
    - id-sd-subject-id(2)
  - id-scp11Role(1) – root for roles used in certificates
    - id-oce(0)
    - id-sd(10)
    - id-kloc-ca(20)
    - id-klcc-ca (30)
    - id-kloc-ka (40)
    - id-klcc-ka (50)