ETSI TS 102 484 V11.1.0 (2013-10)

Smart Cards;
Secure channel between a UICC and an end-point terminal
(Release 11)

Reference
RTS/SCP-T0312vb10

Keywords
security, smart card

ETSI

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - NAF 742 C
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° 7803/88

Important notice

Individual copies of the present document can be downloaded from:
http://www.etsi.org

The present document may be made available in more than one electronic version or in print. In any case of existing or perceived difference in contents between such versions, the reference version is the Portable Document Format (PDF). In case of dispute, the reference shall be the printing on ETSI printers of the PDF version kept on a specific network drive within ETSI Secretariat.

Users of the present document should be aware that the document may be subject to revision or change of status.
Information on the current status of this and other ETSI documents is available at
http://portal.etsi.org/tb/status/status.asp

If you find errors in the present document, please send your comment to one of the following services:
http://portal.etsi.org/chaircor/ETSI_support.asp

Copyright Notification

Contents

Intellectual Property Rights

IPRs essential or potentially essential to the present document may have been declared to ETSI. The information pertaining to these essential IPRs, if any, is publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI Web server (http://ipr.etsi.org).

Pursuant to the ETSI IPR Policy, no investigation, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

Foreword

This Technical Specification (TS) has been produced by ETSI Technical Committee Smart Card Platform (SCP).

The contents of the present document are subject to continuing work within TC SCP and may change following formal TC SCP approval. If TC SCP modifies the contents of the present document, it will then be republished by ETSI with an identifying change of release date and an increase in version number as follows:

Version x.y.z

where:

x the first digit:

- 0 early working draft;
- 1 presented to TC SCP for information;
- 2 presented to TC SCP for approval;
- 3 or greater indicates TC SCP approved document under change control.

y the second digit is incremented for all changes of substance, i.e. technical enhancements, corrections, updates, etc.

z the third digit is incremented when editorial only changes have been incorporated in the document.

1 Scope

The present document specifies the technical implementation of the secure channel requirements specified in TS 102 412 [8].

The present document includes the architecture, functional capabilities and characteristics of the Secure Channel protocol and its associated interfaces transported over the UICC interface specified in TS 102 221 [1] and over the UICC USB interface specified in TS 102 600 [15].

2 References

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the reference document (including any amendments) applies.

- In the case of a reference to a TC SCP document, a non specific reference implicitly refers to the latest version of that document in the same Release as the present document.

Referenced documents which are not found to be publicly available in the expected location might be found at http://docbox.etsi.org/Reference.

NOTE: While any hyperlinks included in this clause were valid at the time of publication ETSI cannot guarantee their long term validity.

2.1 Normative references

The following referenced documents are necessary for the application of the present document.

[1] ETSI TS 102 221: "Smart Cards; UICC-Terminal interface; Physical and logical characteristics".

[2] IETF RFC 4346 (2006): "The Transport Layer Security (TLS) Protocol Version 1.1".

[3] IETF RFC 4366 (2003): "Transport Layer Security (TLS) Extensions".

[4] IETF RFC 4279 (2005): "Pre-Shared Key Ciphersuites for Transport Layer Security (TLS)".

[5] Void.

[6] Void.

[7] ETSI TS 133 110: "Universal Mobile Telecommunications System (UMTS); Key establishment between a UICC and a terminal (3GPP TS 33.110)".

[8] ETSI TS 102 412: "Smart Cards; Smart Card Platform Requirements Stage 1".

[9] ETSI TS 102 223: "Smart Cards; Card Application Toolkit (CAT)".

[10] ETSI TS 124 008: "Digital cellular telecommunications system (Phase 2+); Universal Mobile Telecommunications System (UMTS); LTE; Mobile radio interface Layer 3 specification; Core network protocols; Stage 3 (3GPP TS 24.008)".

[11] IETF RFC 4634 (2006): "US Secure Hash Algorithms (SHA and HMAC-SHA)".

[12] IETF RFC 2104 (1997): "HMAC: Keyed-Hashing for Message Authentication".

[13] FIPS PUB 180-2: "Secure Hash Standard (SHS)".

[14] ETSI TS 102 225 (V7.3.0): "Smart Cards; Secured packet structure for UICC based applications (Release 7)".

[15] ETSI TS 102 600: "Smart Cards; UICC-Terminal interface; Characteristics of the USB interface".

[16] ISO/IEC 9797-1: "Information technology - Security techniques - Message Authentication Codes (MACs) - Part 1: Mechanisms using a block cipher".

[17] Void.

[18] IETF RFC 3268 (2002): "Advanced Encryption Standard (AES) Ciphersuites for Transport Layer Security (TLS)".

[19] IETF RFC 4306 (2005): "Internet Key Exchange (IKEv2) Protocol".

[20] IETF RFC 4301 (2005): "Security Architecture for the Internet Protocol".

[21] IETF RFC 4307 (2005): "Cryptographic Algorithms for Use in the Internet Key Exchange Version 2 (IKEv2)".

[22] Void.

[23] IETF RFC 4303 (2005): "IP Encapsulating Security Payload (ESP)".

[24] ETSI TS 102 483: "Smart cards; UICC-Terminal interface; Internet Protocol connectivity between UICC and terminal".

[25] IETF RFC 4835 (2007): "Cryptographic Algorithm Implementation Requirements for Encapsulating Security Payload (ESP) and Authentication Header (AH)".

[26] IETF RFC 5280: "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile".

[27] IETF RFC 2560: "X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP".

2.2 Informative references

The following referenced documents are not necessary for the application of the present document but they assist the user with regard to a particular subject area.

Not applicable.

3 Definitions and abbreviations

3.1 Definitions

For the purposes of the present document, the following terms and definitions apply:

encrypted blob: Binary Large Object resulting from encrypting plain data

endpoint: application or platform handler on either the terminal or UICC that is capable of being an end of a secure channel

nonce: number used once

NOTE: Random value that is used only once in a cryptographic message to protect against replay attacks.

security association: set of information required for the channel endpoints to start communicating securely

3.2 Abbreviations

For the purposes of the present document, the following abbreviations apply:

3DES	Triple DES
AES	Advanced Encryption Standard
AH	Authentication Header
APDU	Application Protocol Data Unit
ASCII	American Standard Code for Information Interchange
ATR	Answer To Reset
BCD	Binary Coded Decimal
BIP	Bearer Independent Protocol
CBC	Cipher Block Chaining
CL	Counter Limit
CMAC	Cipher-based Message Authentication Code
CRL	Certificate Revocation List
DES	Data Encryption Standard
ESP	Encapsulating Security Payload
FFS	For Further Study
FIPS	Federal Information Processing Standard
GBA	Generic Bootstrap Architecture
HMAC	Hash Message Authentication Code
IETF	Internet Engineering Task Force
IKE	Internet Key Exchange
IKE_SA	Internet Key Exchange Security Association
IP	Internet Protocol
IPsec	Internet Protocol security
MAC	Message Authentication Code
MODP	MODular exPonential
MS	Master Secret
MSA	Master Security Association
OCSP	Online Certificate Status Protocol
PSK	Pre-Shared Key
SA	Security Association
SHA	Secure Hash Algorithm
SSCMAC	Start Secure Channel Message Authentication Code
TCP	Transmission Control Protocol
TLS	Transport Layer Security
TLV	Tag Length Value
TSCA	Terminal-Supported Ciphering Algorithms
TSIM	Terminal-Supported Integrity Mechanisms
UCA	UICC Ciphering Algorithm
UIM	UICC Integrity Mechanism

4 Overview

The present document defines several types of secured data transport protocols that can be used to deliver the secure channel usecases and requirements specified in TS 102 412 [8] and the associated mechanisms that can be used to setup these protocols.

The secured data transport protocols are:

- **TLS - Application to Application:** This protocol secures IP communication between an application in the UICC and an application in the terminal or in a device connected to the terminal. Using this protocol, IP communication by other applications to the UICC may be unsecured. This protocol may be used over the APDU interface specified in TS 102 221 [1] using the BIP - UICC Server mode commands specified in TS 102 223 [9] or over the Ethernet emulation class of the USB interface specified in TS 102 600 [15]. Support for this type of secure channel shall be indicated on per application basis.

- **Secured APDU - Application to Application:** This protocol secures the APDU communication between an application in the UICC and an application in the terminal or in a device connected to the terminal. Using this protocol, APDU communication by other applications to the UICC may be unsecured. This protocol may be used over the APDU interface specified in TS 102 221 [1] or the APDU class of the USB interface specified in TS 102 600 [15]. Support for this type of secure channel shall be indicated on per application basis.

- **IPsec - USB class to USB class:** This protocol secures all IP communication between the UICC and the terminal when TS 102 483 [24] is supported over the Ethernet emulation class of the USB interface specified in TS 102 600 [15].

- **Secured APDU - Platform to Platform:** This protocol secures all APDU communication between the UICC and the terminal. This protocol may be used over the APDU interface specified in TS 102 221 [1] or the APDU class of the USB interface specified in TS 102 600 [15]. Support for this type of secure channel shall be indicated in the ATR.

- **Secured APDU - Platform to Platform CAT:** This protocol secures all CAT APDU communication between the UICC and the terminal and is reusing the Secured APDU - Platform to Platform mechanism but restricted to CAT related APDUs. This protocol may be used over the APDU interface specified in TS 102 221 [1] or the APDU class of the USB interface specified in TS 102 600 [15]. Support for this type of secure channel shall be indicated in the MF file control parameters. A terminal and a UICC claiming conformance to the present document may support this secured data transport protocol.

- **eCAT Secure Channel:** This protocol secures the eCAT communication between the UICC and an eCAT client in the terminal as specified in TS 102 223 [9]. Support for this type of secure channel shall be indicated on per application basis.

Terminals/applications in the terminal/eCAT clients and UICCs/UICC applications claiming conformance to the present document shall explicitly state which of these secured data transport protocols they support.

Secured platform to platform channels between the Terminal and a device connected to the terminal are out of scope of the present document.

To manage the security aspects of these secure channel protocols, Security Contexts are setup which contain security settings and key material. The present document defines four mechanisms to agree key material using:

- **Strong Pre-shared Keys - GBA:** Key material is agreed using the GBA procedures specified in TS 133 110 [7]. The UICC and the terminal shall support this mechanism if GBA is supported.

- **Strong Pre-shared Keys - Proprietary Pre-agreed keys:** These are keys with an entropy of at least 128 bits. The UICC and the terminal may support this mechanism. An eCAT client and the related application on the UICC using the eCAT Secure Channel shall support this mechanism.

- **Weak Pre-shared Keys - Proprietary Pre-agreed keys:** These are keys with an entropy of less than 128 bits (such as password based keys). The UICC and the terminal may support this mechanism.

- **Certificate exchange:** A UICC or a terminal that does not support the GBA mechanism shall support this mechanism. A UICC or a terminal that does support the GBA mechanism may support this mechanism.

5 Secure channel properties

This clause defines common properties for secure channels and details the secure channel lifecycle of each secure channel type defined in the present document.

A secure channel, within the present document, is characterized as having:

- an endpoint on a terminal or connected device;

- an endpoint on a UICC;

- a means of secure bidirectional communication between these endpoints;

- security policy management at each endpoint that prevents insecure communication between these two points.

5.1 Secure Channel Lifecycle

The lifecycle of each secure channel will include the following steps:

- Discovery of support for secure channels by the terminal and the UICC as detailed in the present document.
- Discover endpoints that can communicate securely on the UICC.
- Negotiate secure channel parameters.
- Create a secure channel.
- Communicate over a secure channel.
- Suspend and resume a secure channel.
- Terminate a secure channel.

5.1.1 Secure channel support discovery

Support for the mandatory procedures defined in the present document and support for the Secured APDU - Platform to Platform secure channel shall be indicated in the ATR as defined in TS 102 221 [1].

Support for the Secured APDU - Platform to Platform CAT secure channel shall be indicated in the MF file control parameters as defined in TS 102 221 [1].

A terminal shall only indicate that it supports the procedures in the present document if it is able to execute applications as a trusted platform. Definition of a trusted platform is out of scope of the present document.

5.1.2 Discovery of available endpoints

Each secure channel type defines the mechanisms by which the terminal or the UICC can dynamically discover the available endpoints on the other entity.

NOTE: By their nature, endpoints can dynamically change their availability depending on the activation state of their associated applications.

5.1.3 Negotiate secure channel parameters

For a secure channel to be setup, both ends of the secure channel shall agree on the parameters to be used for this channel. The present document defines these parameters as a "Security Association".

5.1.3.1 Security Associations

A Security Association has the following parameters:

- Identified and authenticated endpoints for both the terminal and the UICC.
- Cryptographic keys.
- Protection algorithms.
- Any additional parameters to be used for securing data transmissions.
- Mechanisms and parameters for identifying secure connections and managing the secure channel.

There are two types of Security Association defined in the present document:

- Master SA.
- Connection SA.

Each secure channel shall have one Master SA and at least one Connection SA.

The terminal and the UICC shall be able to securely store all of the parameters for a minimum of 4 Master SAs and 4 Connections SAs. These Security Association parameters shall not be visible or editable by any process outside of the present document.

5.1.3.2 Master SA

The main security association set up between the channel endpoints is the Master SA. This SA records the following information:

- Channel endpoints.
- Master SA identifier.
- Master SA cryptographic keys (defined as the Master Secret (MS)).
- The algorithms used to establish secure connections.
- Expiration information for the Master SA.

The definition of the Master SA parameters is specific to the type of channel being opened (e.g. Secured APDU - Application to Application).

A Master SA is specific to the endpoints being used and the type of channel being used. If two endpoints need to communicate over a different secure channel type or a secure channel is required to a different endpoint (even if it is on the same device), then a new Master SA shall be used.

The UICC may indicate that there is an existing agreed pre-shared key that can be used to setup this Master_SA.

Master SA's shall exist until they expire or until they are terminated.

A Master SA is used to setup one or more Connection SAs. This enables the setup of a new Connection SA for a secure channel to be negotiated before the current Connection SA expires.

A Master SA shall not be used to directly setup a secure channel.

5.1.3.3 Connection SA

Each Connection SA contains the operational security parameters for a specific secure channel, these parameters are specific to each secure channel type. Connection SAs derive their parameters from a Master SA and have their own lifetime limit.

Connection SAs shall be active until one of the following occurs:

- UICC Power is removed.
- The UICC is reset.
- The Connection SA is terminated.
- The Master SA that the Connection SA is derived from is terminated.
- The terminal determines that the lifetime of the SA has expired.
- The UICC determines that the Connection SA usage counter has reached its limit.

The endpoint may indicate that it supports multiple concurrent connection SAs. The terminal shall use this indication when attempting to setup more than one active Connection SA for that Master SA. For security reasons and for resource consumption optimization, the amount of time during which concurrent Connection SAs exist should be minimized.

5.1.4 Key Agreement

For a Master SA to be setup, a mechanism is needed to share key material between the two endpoints. The present document defines four mechanisms for key agreement that can be used irrespectively as to which secure channel type is used:

- Strong Pre-shared Keys - GBA.
- Strong Pre-shared Keys - Proprietary Pre-agreed keys.
- Weak Pre-shared Keys - Proprietary Pre-agreed keys.
- Certificate exchange.

All pre-shared key agreement mechanisms shall produce values for the following parameters:

- **Ks_local:** This is the secret key used to secure the data transmission between the two endpoints.
- **UICC_ID:** This is a unique identifier for the UICC. This may be the ICCID for the UICC as defined in TS 102 221 [1].
- **UICC_appli_ID:** This is a unique identifier for the UICC application that hosts the UICC endpoint. If Ks_local is intended to be used for 'Secured APDU - Platform to Platform' or 'IPsec - USB class to USB class' secure channel types then UICC_appli_ID shall be set to the ASCII encoded string "platform".
- **Terminal_ID:** This is a unique identifier for the terminal or device connected to the terminal where the terminal endpoint is. This may be the IMEI of the terminal as defined in TS 124 008 [10].
- **Terminal_appli_ID:** This is a unique identifier for the application that hosts the terminal endpoint. If Ks_local is intended to be used for 'Secured APDU - Platform to Platform' or 'IPsec - USB class to USB class' secure channel types then Terminal_appli_ID shall be set to the ASCII encoded string "platform".
- **Weak Key:** This indicates the strength of Ks_local. Weak Key shall be set to 1 if the pre-shared key is based on a low entropy key (i.e. a key of less than 128 bits of entropy such as a user entered PIN or password), otherwise it shall be set to 0.
- **Key Lifetime:** This is the date and time that the key is valid until.
- **Key Counter Limit (CL):** This is the maximum number of times that the key and any derived keys can be used. This is 16 bytes defined as follows:

  Bytes 1 - 2: Reserved for future use.

  Bytes 3 - 4: Number of Master SAs that can be created from this pre-shared key.

  Bytes 5 - 8: Number of Connection SAs that can be derived from each Master SA using this pre-shared key.

  Bytes 9 - 16: Number of individual secure transactions that can be made before the Connection SA, derived from a Master SA using this pre-shared key, shall expire.

The terminal and UICC shall create a value, Ks_Local_Ref, to reference Ks_local where:

Ks_Local_Ref = Terminal_ID || Terminal_appli_ID || UICC_ID || UICC_appli_ID.

The UICC shall reject the setup of a Master SA if the values Terminal_ID and Terminal_appli_ID do not match the intended terminal (and terminal application) that the UICC wishes to securely communicate with.

The terminal shall terminate a Master SA if the values UICC_ID and UICC_appli_ID do not match the intended UICC (and UICC application) that the terminal wishes to securely communicate with.

5.1.4.1 Strong Pre-shared Keys - GBA

A method for establishing a pre-shared key using GBA is defined in TS 133 110 [7].

This method agrees the following values between the UICC and the terminal or connected device:

- A 256 bit shared secret key Ks_local.
- A 10 byte UICC identifier UICC_ID encoded as for ICCID as defined in TS 102 221 [1].
- A 16 byte UICC application identifier UICC_appli_ID (up to 16 bytes).
- A 10 byte terminal identifier Terminal_ID encoded using BCD coding as defined in TS 124 008 [10].
- A terminal application Identifier Terminal_appli_ID (up to 32 bytes).
- A variable length Ks_local Key Lifetime (for use in the terminal).
- A 16 byte Ks_local Counter (for use in the UICC).

For GBA agreed keys, WeakKey shall be set to 0.

Only one GBA key shall be allowed per individual Ks_Local_Ref. If GBA is run again for the same Ks_Local_Ref then the GBA key for that Ks_Local_Ref shall be overwritten by the new key generated and any Master SA or Connection SAs that were setup using the old key shall be terminated.

5.1.4.2 Strong Pre-shared Keys - Proprietary Pre-agreed keys

The terminal and UICC may share strong pre-shared keys (with an entropy of 128 bits or greater) using a proprietary mechanism known to both devices.

The proprietary mechanism used shall agree values for the parameters defined in clause 5.1.4.

5.1.4.3 Weak Pre-shared Keys - Proprietary Pre-agreed keys

The terminal and UICC may share weak pre-shared keys (with an entropy of less than 128 bits) using a proprietary mechanism known to both devices such as password exchange.

The proprietary mechanism used shall agree values for the parameters defined in clause 5.1.4.

Both the UICC and the terminal shall be able to restrict the use of secure channels that are based on a weak pre-shared key.

NOTE: Use of a weak pre-shared key to create a Master SA is not recommended.

5.1.4.4 Certificate exchange

Appropriate key agreement protocols for certificate exchange are defined or referenced as part of the Master SA establishment for each channel type.

5.1.4.5 Expiration values and Counter Limits for Key Material

To maintain the security of keys used by secure channels, all used keys shall have a defined lifetime. The lifetime of a key and a counter limit are set as part of the key agreement and is specific to the strength of the key, how many times it can be used and what it is being used for.

The UICC shall count the following for each pre-shared key:

- The number of Master SAs derived from that key.
- The number of Connection SAs derived from a Master SA.
- The number of transactions handled within a Connection SA.

The UICC shall use the agreed Counter Limit (CL) for each key (as the UICC is not time aware and therefore cannot expire keys using a time-based method) to determine when one of the following conditions has been reached:

- The maximum number of Master SAs have been derived from that pre-shared key. Once this limit is reached the pre-shared key shall be deleted and all Master SAs and Connection SAs based on it shall be terminated by the UICC.
- The maximum number of Connection SAs have been derived from a Master SA. Once this limit is reached the Master SA and Connection SAs based on it shall be terminated by the UICC.
- The maximum number of secure data transactions have occurred for a Connection SA. Once this limit is reached the Connection SA shall be terminated by the UICC.

The terminal or connected device shall have a key lifetime for each key and may have a counter limit for each key. Once either the key lifetime limit or the counter limit is reached for that key, the terminal or connected device shall delete that key and refuse all transactions based on that key. The terminal or connected device shall terminate all security associations that rely on the expired key (including any derived Security Associations).

NOTE: The terminal should not allow the user to manipulate the timing mechanism used to calculate the lifetime of these keys.

5.1.5 Secure Channel Operation

Once a Connection SA has been established (from a Master SA), data may be securely transmitted using the keys and cryptographic algorithms agreed in the Connection SA.

This is specific to each channel type.

5.1.6 Secure Channel Suspension and Resumption

A secure channel shall be considered 'suspended' if all of the Connection SAs for that secure channel have been terminated.

A suspended secure channel shall be resumed when a Connection SA is created using the Master SA for that secure channel.

It is up to the applications using the secure channel to determine the effect that suspension and resumption will have on their communication.

NOTE: It is anticipated that communication over a secure channel are not to be affected by suspension and resumption of a secure channel.

5.1.7 Secure Channel Termination

A secure channel is terminated when the Master SA for that secure channel is terminated. This could either be as a result of a MANAGE SECURE CHANNEL - Terminate secure channel SA command or due to the expiration or erasure of the Master SA key.

A terminated secure channel shall not be able to be restarted, however a new secure channel may be setup to re-establish communication between the two endpoints.

5.2 Use of multiple secure channels

A terminal or UICC conforming to the present document shall be able to support multiple application to application secure channels.

A terminal or UICC conforming to the present document shall be able to support one platform to platform secure channel for each platform secure channel type supported.

A terminal or UICC conforming to the present document shall be able to support multiple application to application secure channels passing through a relevant platform to platform secure channel.

An application to application secure channel shall always pass through a platform to platform secure channel when one exists for that type of channel.

The eCAT clients in a terminal and the UICC may support any number of eCAT Secure Channels.

5.3 Security Policy Enforcement

Once a platform to platform secure channel is established, each platform shall prevent communication, outside of this secure channel, with the other platform.

Once a platform to platform CAT secure channel is established, each platform shall prevent CAT communication, outside of this secure channel, with the other platform.

Once an application to application secure channel is setup, each application that has an endpoint for this channel shall prevent communication, outside of this secure channel and for this type of secure channel, with the other endpoint. It is the responsibility of each application to implement restrictions on other communication channels it may be using, including communication channels that are out of scope of the present document.

Applications on the Terminal or the UICC shall be able to refuse the communication of information with another application if a secure channel is not active between these applications.

The UICC shall be able to refuse any C-APDU if a platform to platform APDU secure channel is required but is not active.

6 TLS - Application to Application lifecycle

This clause defines the lifecycle for an application to application secure channel based on TLS.

Before an application to application TLS secure channel can be initiated, both applications shall be able to communicate over an IP channel using TCP. This may be achieved using following transport layers:

- The ethernet emulation class of the UICC USB interface defined in TS 102 600 [15] together with the IP connectivity layer of TS 102 483 [24].
- The BIP - UICC Server mode commands defined in TS 102 223 [9] using the APDU interface defined in TS 102 221 [1].

6.1 Discovery of available endpoints

For Application to Application TLS secure channel usage, the following mechanisms may be used to identify endpoints that are available:

- The terminal may use Manage Secure Channel APDU - Retrieve UICC Endpoints to discover UICC endpoints.
- The terminal may use Manage Secure Channel APDU - Declare Terminal Endpoints to declare terminal endpoints or endpoints in a connected device, to the UICC. The mechanism that the connected device uses to register its endpoints with the terminal is out of scope of the present document.
- The endpoints may be pre-agreed between the applications on the UICC and the terminal.

6.2 Master SA setup

Both the terminal or the UICC shall be able to initiate a TLS secure channel.

A TLS secure channel shall be initiated by sending a TLS client hello message as defined in RFC 4346 [2]. The Master SA (called a Session in TLS) shall be setup by the TLS handshake protocol defined in RFC 4346 [2].

6.2.1 Setup using a Pre-shared Key

If a pre-shared key (e.g. Ks_local) is used for TLS key establishment then the following shall apply:

- The PSK Identity defined in RFC 4279 [4] shall be set to the value of Ks_Local_Ref.

6.2.2 Setup using Certificates

The UICC shall only accept a terminal certificate, if it verifies correctly and the certificate is signed using an authorized root key recognized by the UICC. In addition the UICC may refuse a certificate if the distinguished name does not correspond to the Terminal_ID (and Terminal_appli_ID if applicable). The UICC and the terminal may verify each other's certificate validity using mechanisms defined in CRL [26] or OCSP [27].

For algorithm negotiation the following TLS cipher suites shall be supported:

- TLS_PSK_WITH_AES_128_CBC_SHA.
- TLS_RSA_WITH_AES_128_CBC_SHA.
- TLS_RSA_PSK_ WITH_AES_128_CBC_SHA.
- TLS_DHE_PSK_WITH_AES_128_CBC_SHA.

The above ciphersuites are defined in RFC 4346 [2], RFC 4279 [4] and RFC 3268 [18].

TLS_PSK_WITH_AES_128_CBC_SHA shall be used when WeakKey=0.

TLS_RSA_WITH_AES_128_CBC_SHA shall be used where the authentication is based on certificates or where there is proprietary additional authentication within the secure channel. The UICC may present a self-signed certificate. The terminal or terminal application should temporarily accept such a certificate during the TLS handshake protocol, if it is able to establish by other means (e.g. successful network authentication) that the handshake protocol is conducted with an authentic UICC.

TLS_RSA_PSK_WITH_AES_128_CBC_SHA or TLS_DHE_PSK_WITH_AES_128_CBC_SHA shall be used when WeakKey=1. The terminal or terminal application shall support both of these cipher suites; the UICC shall support at least one of these cipher suites. If TLS_RSA_PSK_ WITH_AES_128_CBC_SHA is used then the UICC may present a self-signed certificate. The terminal or terminal application should temporarily accept such a certificate during the TLS handshake protocol, if it is able to establish by other means (e.g. successful network authentication) that the handshake protocol is conducted with an authentic UICC.

6.3 Connection SA setup

Connection SAs (called connection states in TLS) are specified in IETF RFC 4346 [2]. These are agreed using the TLS handshake protocol.

6.4 Secure Connection Initiation and Data Transmission

The secure connection initiation and data transmission (TLS record protocol) shall be as defined in RFC 4346 [2]. Secure connections are initiated by the Change Cipher Spec protocol as defined in RFC 4346 [2].

It may be desirable to negotiate a smaller maximum fragment length, as defined in RFC 4366 [3], due to memory limitations or bandwidth limitations. This extension to TLS enables the usage of the following fragment length (when the default value is 2^14):

2^9(1), 2^10(2), 2^11(3), 2^12(4), (255)

- The terminal shall support the Maximum Fragment Length Negotiation as defined in RFC 4366 [3] and shall accept fragment length down to the minimum of 512 bytes.
- The UICC shall support the Maximum Fragment Length Negotiation as defined in RFC 4366 [3] and shall accept fragment length down to the minimum of 512 bytes.

- If the client does not negotiate the Maximum Fragment Length, the server shall accept TLS fragment length with the predefined length of 512 bytes.

The UICC shall maintain a cumulative total of all TLS records sent in each connection state and this total shall not exceed the Transmission counter limit of the connection state (which was obtained from the Master SA). Once the Transmission counter limit is reached, the UICC shall terminate the connection state and no further TLS records shall be transmitted using this Connection SA.

6.5 Secure Connection Termination and Resumption

The TLS secure connection may be terminated by sending a close_notify alert as defined in RFC 4346 [2]. Secure communications can resume by creating a new Connection SA (connection state) using a ChangeCipherSpec message as defined in RFC 4346 [2].

6.6 Master Security Association Termination

There is no explicit way of terminating a Master SA (session) in TLS. However if a session exists and one party wishes to terminate the session, it can refuse to use the session. In this case a completely new session will be established and the previous session is discarded.

7 Secured APDU - Application to Application lifecycle

As the APDU protocol defines the terminal as the master and the UICC as the slave, the APDU secure channel shall only be setup by a terminal application or an application in a device connected to the terminal.

All commands in this clause are defined in TS 102 221 [1].

7.1 Discovery of available endpoints

For Application to Application APDU secure channel usage, the following mechanisms may be used to identify endpoints that are available:

- The terminal application may use Manage Secure Channel APDU - Retrieve UICC Endpoints command to discover UICC endpoints.
- The endpoints may be pre-agreed between the applications on the UICC and the terminal.

7.2 Master SA setup

Application to application secure channel setup shall only be initiated by a terminal application using the Manage Secure Channel APDU - Establish SA - Master SA command. Using the Manage Secure Channel APDU - Establish SA - Master SA command, the terminal application shall supply the Ks_Local_Ref and indicate the supported key agreement mechanisms available for that secure channel.

The UICC application may indicate that a secure channel is required in the MANAGE SECURE CHANNEL - Retrieve UICC Endpoints command or by rejecting an APDU command with the SW1 SW2 set to "Command not allowed - secure channel required".

If the UICC application agrees to the setup request then the UICC application shall respond with a response which includes a 16 byte randomly chosen identifier for the Master SA (MSA_ID) and an indication of which key agreement method it wishes to use from the list of options provided by the terminal application.

If the UICC application rejects the setup request or if there are no available mechanisms for key agreement indicated, then the UICC shall set the SW1 SW2 to 'Execution error - no information given, state of non-volatile memory unchanged' and the Master SA and secure channel procedure shall end.

An Application to application APDU secure channel Master SA may be setup using a pre-shared key or using certificates.

- If a pre-shared key (e.g. from a GBA run) exists and WeakKey=0, then this may be used directly to derive a Master secret for the Master SA. If a pre-shared key exists but WeakKey=1, then a TLS handshake protocol run is required to generate a strong Master secret for the Master SA.

- If no pre-shared key exists but UICC and terminal certificates are available, then the terminal application and UICC application may run a TLS handshake protocol to establish a Master secret for the Master SA.

The terminal application shall instigate the agreed key agreement mechanism.

If a certificate-based key agreement or a weak pre-shared key is to be used for the key agreement then a TLS handshake shall be used to provide key material for the Master SA as follows:

- An IP channel shall be established over the ethernet emulation class of the UICC USB interface defined in TS 102 600 [15] together with the IP connectivity layer of TS 102 483 [24] or over a TCP connection using BIP - UICC Server mode as detailed in TS 102 223 [9] using the TLS port specified in TS 102 483 [24].

- The terminal application sends a 'Client Hello' message to the UICC application to initiate a TLS handshake. The same key agreement algorithms shall be supported as for the Application to Application TLS secure channel.

- The UICC and the terminal may verify each other's certificate validity using mechanisms defined in CRL [26] or OCSP [27]. If verification using one of these mechanisms is performed, the UICC connects to the CRL server or OCSP server using IP connectivity layer of TS 102 483 [24] or using BIP - UICC client mode as defined in TS 102 223 [9]. If a BIP connection is used, the UICC and the terminal need to support 2 simultaneous BIP connections, one in client mode and one in server mode.

- The UICC application and terminal application shall use the 48 byte TLS Master secret (MS_TLS) obtained from the TLS handshake to derive the 256 bit Master secret (MS) of the Master SA as follows: MS = HMAC-SHA-256(MS_TLS, Ks_Local_Ref, MSA_ID). HMAC-SHA-256 is defined in defined in RFC 4634 [11] and FIPS PUB 180-2 [13].

If a strong pre-shared key agreement is indicated, then the UICC application takes the pre-shared key (PSK) referenced by Ks_Local_Ref and derives the Master Secret as MS= HMAC-SHA-256 (PSK,MSA_ID). The terminal application uses the string Ks_Local_Ref to identify the key PSK and then derives the key Master Secret by computing MS=HMAC-SHA-256 (PSK,MSA_ID).

7.3 Connection SA setup

The terminal application shall setup a Connection SA using the Manage Secure Channel APDU - Establish SA - Connection SA command.

The terminal application shall generate a 16 byte Terminal nonce defined as Tnonce.

The terminal application shall send the UICC a Manage Secure Channel APDU - Establish SA - Connection SA command including the MSA_ID, Tnonce, the supported ciphering algorithms for the terminal (TSCA) and the supported integrity mechanisms for the terminal (TSIM).

Upon receipt of the Manage Secure Channel APDU - Establish SA - Connection SA command from the terminal application, the UICC application shall then generate a 16 byte UICC nonce defined as Unonce. The UICC application shall derive 464 bits of key material (KMaterial) from the key MS, and the nonces Unonce and Tnonce as follows: KMaterial = Kexp(MS, Unonce || Tnonce), using the key expansion algorithm KExp as defined in clause 10. The first 128 bits of this key material shall be used as the MAC key K_MAC.

The UICC application replies using a response which includes a randomly generated 16 byte identifier for the Connection SA (CSA_ID), the UICC nonce Unonce, the ciphering algorithm to be used (UCA) and the integrity mechanism (UIM) to be used. This message is protected by the value CSAMAC where CSAMAC = HMAC-SHA-256(K_MAC, MSA_ID || Tnonce || TSCA || TSIM || CSA_ID || Unonce || UCA || UIM) truncated to first 16 bytes as defined in RFC 2104 [12].

The following Ciphering Algorithms shall be supported by the UICC application and the terminal application as a minimum:

- 3DES - outer CBC using 2 keys as defined in TS 102 225 [14].
- 3DES - outer CBC using 3 keys as defined in TS 102 225 [14].
- AES with 128 bit key length in CBC mode with initial chaining value as defined in TS 102 225 [14].

The following integrity mechanisms shall be supported by the UICC application and the terminal application as a minimum:

- CRC32 as defined in TS 102 225 [14].
- ANSI Retail MAC (i.e. MAC algorithm 3 using block cipher DES and padding method 1 as defined in ISO/IEC 9797-1 [16]) without MAC truncation, i.e producing a checksum of 8 bytes length.
- AES with 128 bit key length in CMAC mode as defined in TS 102 225 [14] with a checksum length truncated to the first 64 bits (8 bytes) as output.

Upon receipt of the Establish SA - Connection SA response from the UICC application, the terminal application retrieves the nonce Unonce. The terminal application shall derive 464 bits of key material (KMaterial) from the key MS, and the nonces Unonce and Tnonce as follows: KMaterial = Kexp(MS, Unonce || Tnonce), using the key expansion algorithm KExp as defined in clause 10. The first 128 bits of this key material shall be used as the MAC key K_MAC leaving a remaining 336 bits of key material for ciphering and integrity keys.

The terminal application uses K_MAC to verify the Establish SA - Connection SA response from the UICC application as follows. The terminal application computes CSAMAC' = HMAC-SHA-256(K_MAC, MSA_ID || Tnonce || TSCA || TSIM || CSA_ID || Unonce || UCA || UIM) truncated to the first 16 bytes as defined in RFC 2104 [12]. If CSAMAC' does not equal the value CSAMAC received, then the terminal shall terminate the Connection SA.

If CSAMAC'=CSAMAC, then the terminal application shall send a Manage Secure Channel APDU - Start Secure Channel command to the UICC application containing the secure connection identifier CSA_ID, confirmation of the ciphering algorithm to be used (UCA) and confirmation of the integrity mechanism (UIM). The data in this command is protected by the value SSCMAC where SSCMAC = HMAC-SHA-256(K_MAC, CSA_ID || Unonce || UCA || UIM || CSAMAC) truncated to the first 16 bytes as defined in RFC 2104 [12].

The UICC application uses the key K_MAC to verify the Manage Secure Channel APDU - Start Secure Channel command as follows. The UICC computes SSCMAC' = HMAC-SHA-256(K_MAC, CSA_ID || Unonce || UCA || UIM || CSAMAC) truncated to the first 16 bytes as defined in RFC 2104 [12].

If SSCMAC' does not equal the value SSCMAC sent, then the UICC application shall terminate the Connection SA establishment and set SW1 SW2 to "Authentication error, application specific".

If SSCMAC'=SSCMAC then the UICC application returns the unique secure channel session number to be used for secure data transfer using this Connection SA. This session number is used in the session control within the TRANSACT DATA APDU.

The terminal application and UICC application then derive the ciphering and integrity keys from the remaining 336 bits of KMaterial as follows:

- The ciphering key indicated by KIC shall be taken from the start of the remaining 336 bits of KMaterial. The ciphering key can be at most 168 bits (a 3 key 3DES key), leaving at least 168 remaining bits for the integrity key.
- The integrity key indicated by KID is then taken from the start of the remaining bits left after both the K_MAC and ciphering keys have been taken.

If proprietary algorithms are to be used, then the keys for these algorithms may either be derived from the 336 bits of key material available, or proprietary mechanisms can be used to determine how to locate the keys indicated by KIC and KID.

7.4 Secure Connection Initiation and Data Transmission

Once a Manage Secure Channel APDU - Start SecureChannel command has been received by the UICC application and acknowledged, the UICC application and terminal application can initiate their security policy and start to secure transmitted data.

To send and receive APDUs securely through the APDU secure channel, the terminal application shall use the Transact Data APDU command defined in TS 102 221 [1] with the P1 parameter set to the value returned in the response to the Manage Secure Channel APDU - Start Secure Channel command. The coding of encrypted APDUs exchanged using the Transact Data command is described in clause 10 "Encrypted Data coding".

The terminal application and UICC application shall handle the encryption/decryption of APDUs, and their responses, with up to 255 bytes of data using the secure channel segmentation detailed in TS 102 221 [1].

Each encrypted message, in either direction, shall have its own 8 bytes transaction counter value that shall be the last successful message counter value + 1. This transaction counter is incremented regardless of execution errors or aborted transactions. The same transaction counter shall be used for both directions of communication. The transaction counter is reset when a new Connection SA is established.

On receipt of encrypted blobs, the terminal application or UICC application receiving the blob shall:

- Re-assemble the encrypted blobs.
- Decrypt the combined encrypted blob using the keys and mechanisms agreed for that secure channel.
- Verify that the message is valid by checking the integrity protection.
- Check that the counter is valid.

If the message is valid then the terminal application or UICC application that has decoded the message shall action the APDU or APDU response.

If the message is invalid then the terminal application or UICC application that has decoded the message shall not action the APDU or APDU response.

7.5 SA Termination and Resumption

To terminate an existing APDU secure channel Master or Connection SA, the terminal application shall use the Manage Secure Channel APDU - Terminate secure channel SA command defined in TS 102 221 [1].

If a Connection SA is indicated then the MAC value shall be set to the first 16 bytes of HMAC SHA 256(K_MAC, CSA_ID).

If a Master SA is indicated then the MAC value shall be set to the first 16 bytes of HMAC SHA 256(MS, MSA_ID).

The UICC application shall acknowledge the Manage Secure Channel APDU - Terminate secure channel SA command with a status word indicating success or failure.

The UICC application can indicate that a key or a security association has expired using the SW1 SW2 response "Security session or association expired".

In order to resume a secure channel, the terminal application shall start a completely new Connection SA using the Manage Secure Channel APDU - Establish SA - Connection SA command.

8 Ipsec - USB class to USB class lifecycle

This clause defines the USB class to USB class secure channel for the USB Ethernet emulation class specified in TS 102 600 [15] using Ipsec.

Before an Ipsec - USB class to USB class secure channel can be initiated, both applications shall be able to communicate using IP connectivity as specified in TS 102 483 [24] over the USB Ethernet emulation class specified in TS 102 600 [15].

8.1 Discovery of available endpoints

If Ipsec is supported, then both the terminal or the UICC shall be able to initiate the Ipsec secure channel as defined in RFC 4301 [20].

Endpoint discovery shall use the mechanism defined in clause 8.4.

8.2 Master Security Association

Ipsec specifies the establishment of its own Master Sas. These are called IKE Sas in Ipsec. An IKE SA shall be setup using the IKE v2 protocol defined in RFC 4306 [19]. Both pre-shared key and certificate based key agreement mechanisms in IKE v2 shall be supported.

The IKE v2 protocol is initiated by sending an IKE v2 IKE_SA_INIT message as defined in RFC 4306 [19]. When running IKE v2, the initiator and responder identifiers (Idi and Idr) shall be as follows:

If the terminal is the initiator and the UICC the responder then:

- Idi is the string Terminal_ID || Terminal_appli_ID.
- Idr is the string UICC_ID || UICC_appli_ID.

If the UICC is the initiator and the terminal is the responder then:

- Idi is the string UICC_ID || UICC_appli_ID.
- Idr is the string Terminal_ID || Terminal_appli_ID.

If a pre-shared key is to be used to establish the IKE SA, then the UICC shall use the key corresponding to Ks_Local_Ref (this corresponds to Idi || Idr if the UICC is the initiator or Idr || Idi if the UICC is the responder), and the terminal shall use the key corresponding to Term_label (this corresponds to Idi || Idr if the terminal is the initiator or Idr || Idi if the terminal is the responder).

The mandatory algorithms for IKE v2 defined in RFC 4307 [21] shall be supported. In addition, the 2 048 MODP Group for Diffie Helman, and the ENCR_AES_CBC algorithm for Transform Type 1 (both marked as SHOULD+ in RFC 4307 [21]) shall be supported.

If a pre-shared key is used to derive the Master SA, then the number of Master Sas that can be derived from the pre-shared key is bounded by the Master CL associated with the pre-shared key. If none exists, then a default counter value shall be associated to the pre-shared key. If certificates are used, then default values for the Connection CL and Transmission CL shall be set.

8.3 Secure Connections

Ipsec supports the establishment of Connection Sas. These are called Child Sas in Ipsec and are established from the IKE SA using the Create Child SA command.

Ipsec - **USB class to USB class** secure channel shall use ESP in tunnel mode as defined in RFC 4303 [23].

The mandatory cryptographic algorithms for Ipsec AH and ESP defined in RFC 4835 [25] shall be supported.

The number of Child Sas that can be derived from the Master SA is bounded by the Connection CL.

Assignment of IP addresses for an Ipsec secure channel is FFS.

8.4 Secure Connection Initiation and Data Transmission

Ipsec secure channel communication shall be as defined in RFC 4301 [20] and RFC 4303 [23].

A terminal or UICC requiring an Ipsec - **USB class to USB class** secure channel shall initiate the Ipsec secure channel. If secure channel setup fails (e.g. because it is not supported by the other side), either all IP traffic shall be blocked or traffic from and to applications requiring the Ipsec secure channel shall be blocked.

The number of IP packets that can be transmitted using a Child SA is bounded by the Transmission CL.

8.5 Secure Connection Termination and Resumption

A Child SA may be terminated by erasing the Child SA if either side wishes to terminate the connection (e.g. a timeout occurs or some expiry condition is reached). Secure communications can be resumed by using the Create Child SA command to create a new Child SA from the IKE SA.

Re-keying of Child Sas (i.e. establishment of a new equivalent Child SA from the IKE SA before the old Child SA terminates) as defined in RFC 4306 [19] shall be supported in order to prevent breaks in transmission. A Child SA can be re-keyed as long as the number of Child Sas derived from the IKE SA does not exceed the Connection CL.

8.6 Master Security Association Termination

The IKE SA may be terminated by deleting the IKE SA if either side wishes to terminate the IKE SA (e.g. due to expiration of Lifetimes or Counters). Re-keying of an IKE_SA using the Create Child SA command as defined in RFC 4306 [19] shall not be supported. If an IKE SA terminates, then a completely new IKE SA shall be created from existing pre-shared key material or certificates.

9 Platform to Platform APDU secure channel lifecycle

9.1 Platform to Platform APDU secure channel

A UICC shall indicate that it requires a Platform to Platform APDU secure channel in the ATR as defined in TS 102 221 [1].

If the UICC indicates a Secured APDU - Platform to Platform secure channel is required and if the terminal supports Secured APDU - Platform to Platform secure channel, then the terminal shall negotiate the Secured APDU - Platform to Platform secure channel immediately following the PPS procedure.

If the UICC indicates a Secured APDU - Platform to Platform secure channel is required and if the terminal does not support Secured APDU - Platform to Platform secure channel or if there are no available mechanisms for key agreement between the terminal and the UICC, then the UICC shall allow the non-secured APDU commands.

A Platform to Platform APDU secure channel shall use the processes defined for an application to application APDU secure channel with the UICC and terminal application identifiers set to the ASCII encoded string "platform".

If the UICC requires a Platform to Platform APDU secure channel, then it shall only process the following APDUs outside of the secure channel.

Table 9.1: Allowed APDUs outside of a secure channel when platform to platform APDU secure channel is required

GET RESPONSE
MANAGE SECURE CHANNEL
TRANSACT SECURE DATA

The Platform to platform APDU channel shall only be set up on logical channel 0, and while a platform to platform secure channel is active, the UICC shall reject APDUs on any other logical channel. It shall be possible to use logical channels in the encrypted APDU payload.

9.2 Platform to Platform CAT APDU secure channel

A UICC shall indicate that it requires a Platform to Platform CAT APDU secure channel in the MF file control parameters as defined in TS 102 221 [1].

If the UICC indicates a Secured APDU - Platform to Platform CAT secure channel is required and if the terminal supports Secured APDU - Platform to Platform CAT secure channel, then the terminal shall negotiate the Secured APDU - Platform to Platform CAT secure channel after the MF is selected and before any CAT command is issued.

If the UICC indicates a Secured APDU - Platform to Platform CAT secure channel is required and if the terminal does not support Secured APDU - Platform to Platform CAT secure channel or if there are no available mechanisms for key agreement between the terminal and the UICC, then the UICC may allow the non-secured APDU commands.

A Platform to Platform CAT APDU secure channel shall use the processes defined for an application to application APDU secure channel with the UICC and terminal application identifiers set to the ASCII encoded string "CAT".

If the UICC requires a Platform to Platform APDU CAT secure channel, then it shall not process the CAT APDUs (as defined in TS 102 221 [1]) outside of the secure channel.

Table 9.2: Forbidden APDUs outside of a secure channel when platform to platform CAT APDU secure channel is required

TERMINAL PROFILE
ENVELOPE
FETCH
TERMINAL RESPONSE
NOTE: The triggering of a CAT session with the Status Word '91 XX' may occur inside or outside of the secure channel.

10 Encrypted data coding

This clause applies to Application to Application APDU secure channel and Platform to Platform APDU secure channel. For an eCAT secure channel, the coding of secured eCAT is defined in TS 102 223 [9].

Data to be sent and its response is encrypted together with a nonce, a counter, padding and a checksum. The padding length shall be chosen so that the data to be encrypted is a multiple of the block size for the algorithm used. The padding length may be larger than the algorithm block size to disguise the length of the APDU being sent or the response being received.

Encrypted data is sent using the TRANSACT DATA command as described in TS 102 221 [1]. The encrypted data is sent in encrypted data TLV objects.

For each secure channel, TRANSACT DATA APDUs with encrypted data TLV objects shall always contain fixed number of bytes of data. If the data is sent using several APDUs, each of the APDUs, including the last one, shall contain the same fixed number of bytes of data. This data size is indicated in the endpoint discovery mechanism for each secure channel. If necessary, the TRANSACT DATA command data shall be padded to create the correct length of message as described below.

If the UICC sends back an encrypted data TLV object, the response data shall always be the same fixed number of bytes as indicated in the endpoint discovery mechanism for each secure channel. If the data is sent using several APDUs, each of the APDUs, including the last one, shall contain the same fixed number of bytes of data. If necessary, the TRANSACT DATA command data shall be padded to create the correct length of message as described below. The data may be C-APDUs and R-APDUs or any other content agreed between the terminal and the UICC.

In case of application-to-application secure channel, and if the encrypted data is C-APDUs and R-APDUs, then the status words SW1 and SW2 which are part of the R-APDU are only destined for the receiving terminal applications and are not interpreted on the physical interface. As an example, status words 91 XX are meaningless as status words in an encrypted R-APDU.

10.1 Mapping Data from the Terminal to the UICC

10.1.1 Structure of the data to be encrypted

This clause describes the structure of the data that, once encrypted, forms the value part of the encrypted blob TLV to be sent to the UICC.

Table 10.1: Coding of the Data to be encrypted

Byte(s)	Description	Length
1 to 8	Nonce	8
9 to 16	Counter	8
16 to 16+x	APDU BER-TLV	x
17+x to 17+x+y	Padding	y
18+x+y to 25+x+y	Checksum	8

Nonce:

- Coding: 8 byte random value.

Counter:

- Contents: transaction counter as described in clause 7.4.
- Coding: 8 byte hex value.

APDU BER-TLV:

- Contents: TLV containing the APDU to be sent to the UICC.
- Coding: see table 10.2.

Table 10.2: Coding of the APDU BER-TLV object

Byte(s)	Description	Value	Length
1	Tag	'82'	1
2 (or 2 to 3)	Length		1 or 2
3 to 3+a (or 4 to 4+a)	APDU		a

Padding:

- Contents: non deterministic padding data. The length of the padding data shall be chosen such that the total length of the data to be encrypted is a multiple of the block size of the ciphering algorithm to be used.
- Coding: hex bytes.

Checksum:

- This is the 8 byte checksum. This checksum is calculated using the nonce, counter, APDU BER-TLV and padding in the order described in table 10.1. The checksum algorithm used shall be the agreed integrity mechanism for this secure channel.

10.1.2 Definition of the encrypted blob TLV when sending data to the UICC

This clause defines the encrypted blob TLV when it is used to transfer encrypted data to the UICC.

Table 10.3: Coding of the Encrypted Data BER-TLV object

Byte(s)	Description	Value	Length
1	Tag	'81'	1
2	Length		1, 2 or 3
3	Encrypted Data		x

Encrypted Data:

- Contents: This shall be the result of the encryption of the data to be encrypted defined in clause 10.1.1 using the agreed encryption algorithm.
- Coding: hex bytes.

10.1.3 Mapping of the encrypted blob TLV to C-APDUs

The encrypted blob shall be transported as 1 or more TRANSACT DATA commands. If more than 1 TRANSACT DATA command is required to transport the message then the message shall be split so that the tag and length are only present in the first message. The length of each TRANSACT DATA command data shall be the agreed container size for this secure channel. As the Encrypted Blob TLV may not be an exact multiple of the TRANSACT DATA container size, the remaining bytes of the last TRANSACT DATA command data shall be padded with '00'. This padding shall not be included in the calculation of the length for the encrypted blob TLV. Figure 10.1 shows an example of this split where three TRANSACT DATA commands are used.

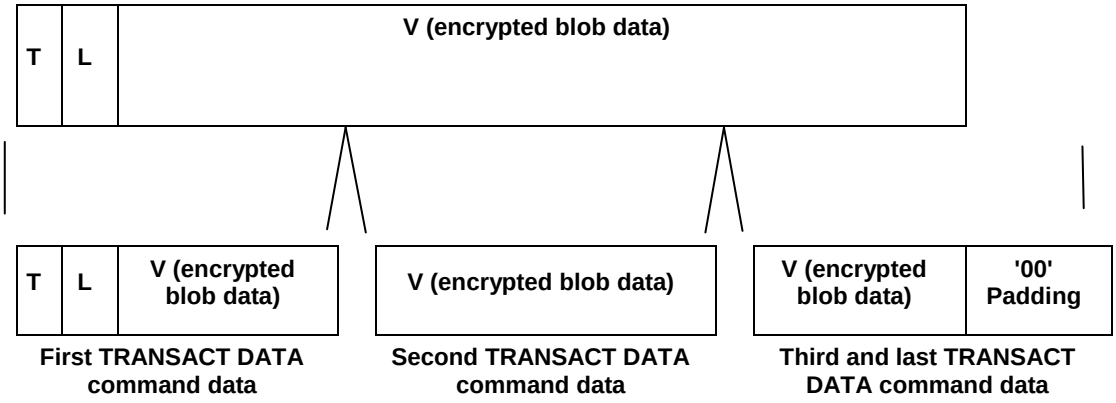

Figure 10.1: An example showing the splitting of an encrypted blob TLV into three TRANSACT DATA commands.

10.2 Mapping response from the UICC to the Terminal

10.2.1 Structure of the data to be encrypted

This clause describes the structure of the data that, once encrypted, forms the value part of the encrypted blob TLV to be sent to the UICC.

Table 10.4: Coding of the Data to be encrypted

Byte(s)	Description	Length
1 to 8	Nonce	8
9 to 16	Counter	8
16 to 16+x	APDU Response BER-TLV	x
17+x to 17+x+y	Padding	y
18+x+y to 25+x+y	Checksum	8

Nonce:

- Coding: 8 byte random value.

Counter:

- Contents: transaction counter as described in clause 7.4.
- Coding: 8 byte hex value.

APDU Response BER-TLV:

- Contents: TLV containing the APDU to be sent to the UICC.
- Coding: see table 10.5.

Table 10.5: Coding of the APDU Response BER-TLV object

Byte(s)	Description	Value	Length
1	Tag	'83'	1
2 (or 2 to 3)	Length		1 or 2
3 to 3+a (or 4 to 4+a)	APDU Response		a

Padding:

- Contents: non deterministic padding data. The length of the padding data shall be chosen such that the total length of the data to be encrypted is a multiple of the block size of the ciphering algorithm to be used.
- Coding: hex bytes.

Checksum:

- This is the 8 byte checksum. This checksum is calculated using the nonce, counter, APDU Response TLV and padding in the order described in table 10.4. The checksum algorithm used shall be the agreed integrity mechanism for this secure channel.

10.2.2 Definition of the encrypted blob TLV when receiving data from the UICC

This clause defines the encrypted blob TLV when it is used to transfer encrypted data from the UICC.

Table 10.6: Coding of the Encrypted Data BER-TLV object

Byte(s)	Description	Value	Length
1	Tag	'81'	1
2	Length		1, 2 or 3
3	Encrypted Data		x

Encrypted Data:

- Contents: This shall be the result of the encryption of the data to be encrypted defined in clause 10.2.1 using the agreed encryption algorithm.
- Coding: hex bytes.

10.2.3 Mapping of the encrypted blob TLV to C-APDUs

The mapping of the response encrypted Blob TLV to the responses of C-APDUs shall be the same as for the mapping of encrypted blob TLVs to C-APDUs described in clause 10.1.3.

11 Key Expansion Function Definition

The key expansion function Kexp is based on the Key Expansion function defined in IKE v2 (RFC 4306 [19]) and is designed to produce any required amount of key material from a single cryptographic key. In order to do this, the HMAC-SHA-256 algorithm, which produces output of 256 bits is used iteratively until enough key material is available.

For input a key K and an arbitrary length string str, the function Kexp produces a stream of 256 bit output strings T1, T2, T3, etc using HMAC-SHA-256 as follows:

$$Kexp(K,str) = T1 \,||\, T2 \,||\, T3 \,||\, \ldots$$

Where:

$$T1 = HMAC\text{-}SHA\text{-}256(K, str \,||\, 0x01)$$

$$T2 = HMAC\text{-}SHA\text{-}256(K, T1 \,||\, str \,||\, 0x02)$$

$$T3 = HMAC\text{-}SHA\text{-}256(K, T2 \,||\, str \,||\, 0x03)$$

And so on until enough key material has been produced.

Key material of the desired length (e.g. 464 bits are required for Kmaterial in clause 7.3) is taken from the output key stream of Kexp.

EXAMPLE: If 464 bits of key material (Kmaterial) are required (as is the case in clause 7.3), then Kmaterial will consist of T1 concatenated with the first 208 bits of T2. In this case, only 2 iterations of HMAC-SHA-256 are required to generate the required key material.

12 eCAT Secure Channel

This clause defines the life cycle for an eCAT Secure Channel between an eCAT client in the terminal and the UICC.

In case the terminal supports CAT over modem interface, see TS 102 223 [9], the eCAT client may reside in the modem or in the Connected Entity.

For an eCAT secure channel, the data coding of secured eCAT is defined in TS 102 223 [9].

Data structures defined in clause 7 for the APDU application to application secure channel are re-used as far as possible for the eCAT secure channel.

12.1 Discovery of available endpoints

Discovery of available endpoints is described in TS 102 223 [9].

12.2 Master SA setup

Only the mechanisms required for proprietary pre agreed strong pre-shared keys are specified in this version of the document for the eCAT secure channel.

The setup of the eCAT secure channel shall be initiated by the UICC using ENCAPSULATED SESSION CONTROL (request master SA setup).

The command data shall include:

- Term label - UICC_appli_ID

The terminal response from the eCAT client shall include:

- Key Agreement Mechanism
- Term label - Terminal_ID
- Term label - Terminal_appli_ID
- Term label - UICC_Identifier
- Term label - UICC_appli_ID

Terminal_appli_ID should be set to the UTF-8 encoded eCAT client name specified in TS 102 223 [9].

The UICC shall generate an MSA_ID and calculate the Master Secret MS as defined in clause 7.2.

If the UICC_appli_ID does not match the intended UICC application that the eCAT client wishes to securely communicate with, the eCAT client shall set the result code of the terminal response to "Command data not understood by terminal" and shall not include the SA_template in the terminal response. This shall terminate the secure channel setup.

If the values Terminal_ID and Terminal_appli_ID do not match the intended terminal and terminal application/eCAT client that the UICC wishes to securely communicate with, the UICC shall close Master SA.

12.3 Connection SA setup

For eCAT, only one Connection SA can exist at any point in time per eCAT client.

The setup of a Connection SA shall be initiated by the UICC using ENCAPSULATED SESSION CONTROL (request Connection SA setup).

The command data shall include:

- Key Agreement Mechanism
- MSA_ID

Upon receipt of the command, the eCAT client shall generate a 16 byte terminal nonce defined as Tnonce.

The terminal response from the eCAT client shall include:

- Algorithm and integrity

- MSA_ID
- Tnonce

The UICC shall select an integrity mechanism and optionally a ciphering mechanism, generate the Unonce and the CSA_ID, derive key material and calculate the CSAMAC as defined in clause 7.3.

The following Ciphering Algorithms shall be supported:

- AES with 128 bit key length in CBC mode with initial chaining value as defined in TS 102 225 [14].

Encryption shall always be accompanied by an integrity mechanism.

The following integrity mechanisms shall be supported:

- AES with 128 bit key length in CMAC mode as defined in TS 102 225 [14] with a checksum length truncated to the first 64 bits (8 bytes) as output.

The UICC shall then send an ENCAPSULATED SESSION CONTROL (request Secure Channel Start) to the eCAT client, containing:

- Algorithm and integrity
- CSA_ID
- Unonce
- CSAMAC

The eCAT client shall derive key material and calculate the SSCMAC as defined in clause 7.3.

The terminal response from the eCAT client shall include:

- Algorithm and integrity
- CSA_ID
- SSCMAC
- Endpoint data container size

Endpoint data container size shall be empty as it is not used for eCAT.

As only one Connection SA exists at any point in time per eCAT client, no session number is used for eCAT.

Ciphering and integrity keys shall be derived from KMaterial as defined in clause 7.3.

12.4 Secure Connection Initiation and Data Transmission

After a successful ENCAPSULATED SESSION CONTROL (request Secure Channel Start), all subsequent eCAT commands shall be protected with the indicated security mechanism.

To also protect the eCAT client's profile, the first command after establishment of the secure channel shall be a protected Encapsulated Profile. If any other command is sent by the eCAT client after establishment of the secure channel, the UICC shall close the secure connection.

If the eCAT client wants to send a protected ENVELOPE CONTAINER (TERMINAL APPLICATIONS), this shall be done after the protected Encapsulated Profile.

12.5 SA Termination and Resumption

An ENCAPSULATED SESSION CONTROL command shall be used to end a secure session. Closing the encapsulated command session shall also end the secure session.

Annex A (informative): Change history

This annex lists all changes requests (CR) applied to the present document.

Change history								
Date	**Meeting**	**Plenary Doc**	**CR**	**Rev**	**Cat**	**Subject/Comment**	**Old**	**New**
2008-01	SCP-35	SCP-080045	001		B	Coding of secure data for secure channel over APDUs	7.0.0	7.1.0
2008-07	SCP-38	SCP-080371	002	1	F	Corrections and clarifications on Secure Channel	7.1.0	7.2.0
2008-07	SCP-38	SCP-080359	004	1	F	Removal of Ipsec indication	7.1.0	7.2.0
2008-07	SCP-38	SCPt080320	005		F	Correct RFC references	7.1.0	7.2.0
2008-07	SCP-38	SCP-080360	006	1	F	Specify selected options for Ipsec	7.1.0	7.2.0
2008-10	SCP-39	SCP-080463	007		F	Essential correction to the structure of the encrypted blob	7.2.0	7.3.0
2009-05	SCP-41	SCP-090119	009		F	Correction on Pre-Shared Keys - GBA	7.3.0	7.4.0
2009-05	SCP-41	SCP-090138	008		F	Correction to TS 102 484 mapping of data and clarification of container size.	7.3.0	7.4.0
2009-07	SCP-42	SCP-090261	010	1	F	Alignment of encrypted data BER TLV tag with TS 101 220.	7.4.0	7.5.0
2009-10	SCP-43	SCP-090328	012		F	Transaction counter behaviour on aborted transaction	7.5.0	7.6.0
2009-10	SCP-43	--------	---	-	-	Creation of Release 8 of the specification for the purpose of creating Release 9	7.6.0	8.0.0
2009-10	SCP-43	SCP-090328	011		B	Enhancements on security algorithms	8.0.0	9.0.0
2010-07	SCP-45	SCP(10)0178	019	1	A	TERMINATE SA behaviour for application-to-application Secure Channel	9.0.0	9.1.0
2010-10	SCP-46	SCP(10)0265	023	-	A	Correction to AUTHENTICATE command handling in P2P Secure channel	9.1.0	9.2.0
2010-10	SCP-46	SCP(10)0294	020	-	C	Security associations behaviour for A2A Secure Channel	9.2.0	10.0.0
2010-10	SCP-47	SCP(11)0059r1	024	1	B	Clarification on certificate validity check during TLS handshake	10.0.0	10.1.0
2011-09	SCP-52	SCP(11)0282r1	025	-	D	Wrong reference to ICCID_ID in section 5.4.1	10.1.0	11.0.0
2012-12	SCP-57	SCP(12)000265r1	025	1	B	Secure Channel for encapsulated CAT	11.0.0	11.1.0
2012-12	SCP-57	SCP(12)000267	027	-	B	Securing CAT communications	11.0.0	11.1.0

History

Document history		
V11.0.0	September 2012	Publication
V11.1.0	October 2013	Publication