ERTMS/ETCS
On-line Key Management FFFIS
REF : SUBSET-137 ISSUE : 4.0.0 DATE : 05-07-2023

Company	Technical Approval	Management approval
ALSTOM		
AZD		
CAF		
HITACHI RAIL STS		
MERMEC		
SIEMENS		
THALES		

1. MODIFICATION HISTORY

Issue Number Date	Section Number	Modification / Description	Author
0.1.0 20-05-2015	All	Release for review	KMS WG
1.0.0 17-12-2015	-	Baseline 3 2nd release version	PP
1.0.0.1_CR1359 09-05-2021	Move Section SECURITY INTERFACE SPECIFICATIONS to Subset-146	In regards to Baseline SS-137-v1.0.0 the following section (security interface) were moved to new SS-146 Security Layers for ETCS Applications 4.2.1.1 Figure 1 4.2.1.6 4.3.1.1 4.3.1.2-4 4.3.1.4.1 4.3.1.5-6 4.3.1.7 4.3.1.8 4.3.1.9 4.3.1.0 4.3.1.11 4.3.1.12 4.3.1.13 4.3.1.14 4.3.3 4.4 6.1 6.2 6.2.2.2 6.2.2.3 6.2.2.4 6.2.2.5 6.2.2.6 6.2.3.1 6.2.3.2 6.2.3.3 6.2.3.4	JM

		6.2.3.5-6 6.2.3.7 6.2.3.8 6.2.3.9 6.2.4.1 6.2.4.2 6.2.4.3-4 6.2.4.5 6.2.4.6 6.2.4.7.2 6.2.4.7.3 6.2.4.7.4 6.2.4.7.5 6.2.4.7.6 6.2.4.7.7 6.2.4.8.1 6.2.4.9 6.2.4.9.1 6.2.4.9.2 6.2.4.9.3 6.2.4.10 6.2.4.11 6.2.4.12 6.2.4.13 6.2.4.14 6.2.4.15 6.2.4.16 6.3.1 6.3.2 6.3.3 6.4	
1.0.0.1 24-01-2020	§ 5.3.17, 5.3.4, 5.5.3.5, 5.3.1.3, 5.3.12, 5.4.4.5, 4.3.1.10, 5.6.1.6	Updated due to CR1307 editorial improvement	MW
1.0.0.2 30-01-2020	Appendix A, Example 2	Updated due to CR1307 editorial improvement	MP
1.0.0.3 03-03-2020	§ 5.3.17, 5.3.4, 5.5.3.5, 5.3.1.3, 5.3.12, 5.4.4.5,	Updated due to CR1307 editorial improvement	MW

	4.3.1.10, 5.6.1.6, Annex A,		
1.0.0.4 05-02-2020	5.3.1.5	typo	MW
1.0.0.5 19-05-2020	3.2	Updated due to CR1307 editorial improvement References [RFC-2560]; [RFC-6277] replaced by [RFC-6960]	MW
1.0.0.6 25-06-2020	5.3.15	Updated due to CR1307 editorial improvement	MW, MK
1.0.0.7 13-07-2020		Updated due to CR1307 editorial improvement	
1.0.0.8 14-10-2020		Left intentionally free (now in SS-146)	
1.0.0.9 01-06-2022	Merged CR1307, CR1359, CR1415	Added CR reference by use of comments	MW, JM
1.0.0.10 22-06-2022	history Security Interface 7.2	Clean-up Re-established Chapter 6.2 in respect to Baseline SS-137-1.0.0 Dummy to keep numbering aligned	MW
3.9.2 23-02-2023	sections 5.3.17, 5.3.4, 5.6.1.6, 5.3.15,	CR 1307 changes reassigned to CR1428	PL
3.9.3 31-05-2023	Sections 5.3.15, 5.6.1.6	EECT comments Outcome of B4R1 3rd consolidation phase	PL
3.9.4 02-07-2023	Template/footer SS-146 reference name	EECT comments +update to 4.4.1.1 Outcome of B4R1 4th consolidation phase	PL
4.0.0 05-07-2023	-	Baseline 4 1st release version	GP

2. TABLE OF CONTENTS

3. INTRODUCTION

3.1 Scope and Purpose

3.1.1.1 ERTMS/ETCS applications use open transmission systems to transfer messages between ERTMS/ETCS equipment.

3.1.1.2 Data transmission links implemented over open transmission systems are inherently vulnerable as unauthorised access cannot be excluded. Therefore, it is important to guarantee the integrity and authentication of messages sent over a non-trusted transmission medium. ERTMS/ETCS applications use cryptographic techniques with secret keys to achieve this.

3.1.1.3 ERTMS/ETCS specifications, such as [Subset-037] and [Subset-098], assume that the cryptographic keys are already installed in the equipment. However, they do not describe how and in which format these keys are transferred from the source (a Key Management Centre) to the destination (a KMAC entity), and how they are installed.

3.1.1.4 This Subset specifies a Key Management System which covers the management of on-line distribution of cryptographic keys between Key Management Centres and from a Key Management Centre to KMAC entities.

3.1.1.5 The harmonisation of these interfaces is done in a policy-open way, allowing each operator to implement a key management policy adequate for their specific security needs; e.g. using different authentication keys for each pair of KMAC entities, or using the same authentication key for a group of KMAC trackside entities.

3.1.1.6 This Subset is applicable for all KMAC entities whose communication is based on cryptographic keys and therefore need to provide an interface for installation, update and deletion of such keys.

3.1.1.7 This Subset is also applicable for Key Management Centres performing key management tasks for KMAC entities.

3.2 References

[ENISA]	Algorithms, key size and parameters report 2014	November 2014
[EN-50159]	Safety-related communication in transmission systems	September 2010
[RFC-1320]	The MD4 Message-Digest Algorithm	April 1992
[EIRENE SRS]	GSM-R System requirements specification	
[Subset-023]	ERTMS/ETCS; Glossary of Terms and Abbreviations	
[Subset-037]	ERTMS/ETCS; EuroRadio FIS	
[Subset-038]	ERTMS/ETCS; Off-line Key Management FIS	
[Subset-098]	ERTMS/ETCS; RBC-RBC Safe Communication Interface	
[Subset-114]	ERTMS/ETCS; KMC-ETCS Entity Off-line KM FIS	
[Subset-146]	ERTMS End-to-End Security	

3.3 Acronyms and Abbreviations

3.3.1.1 For general abbreviations refer to [Subset-023]. Additional abbreviations relevant for key management and used in this document are specified here.

Abbreviation	Definition
DB	DataBase
DN	Distinguished Name
TLS	Transport Layer Security
UTF-8	Unicode Transformation Format 8-bit

3.4 Terms and Definitions

3.4.1.1 For general terms refer to [Subset-023]. Additional terms relevant for key management and used in this document are specified here.

Term	Definition
ETCS entity	ETCS EVC, RBC or RIU
Expanded ETCS ID	The unique identifier of a KMS entity, consisting of its ETCS ID type and its ETCS ID
Key database	Contains the key entries in the KMS entities (note: the term “database” is used here for any method of storing key entries)
Key entry	An authentication key (KMAC) with the following related information: ⇨ identifier of the KMC that issued the key ⇨ key serial number ⇨ key validity period ⇨ list of KMAC entities to which this key is allocated
Key serial number	The number uniquely identifying one key within the set of keys generated by a KMC
KMAC entity	KMAC on-board entity or KMAC trackside entity
KMAC on-board entity	ETCS on-board equipment
KMAC trackside entity	RBC or RIU
KMS entity	KMAC entity or KMC
Pseudorandom number generator	A pseudorandom number generator is an algorithm for generating a sequence of numbers whose properties approximate the properties of sequences of random numbers.
Transaction	Message from a KMS entity requiring a response from the peer entity and the response to this message from the peer entity

4. KEY MANAGEMENT PRINCIPLES AND CONCEPTS

4.1 Introduction

4.1.1.1 In order to secure the communication over a Category 3 [EN-50159] open transmission system, the on-board and trackside equipment in the ERTMS/ETCS system exchange information using the EuroRadio protocol [Subset-037].

4.1.1.2 When an ETCS equipment establishes a connection with another ETCS equipment (e.g. between an EVC and an RBC), both must be able to authenticate the other equipment and verify that it is an authorised entity. That way, the authenticity and integrity of the information exchanged between them is also achieved.

4.1.1.3 The method for authenticating both communicating entities is based on an Identification and Authentication (I&A) dialogue. In order to ensure protection, this dialogue shall take place each time two entities start a new safe connection.

4.1.1.4 After a successful I&A dialogue, data is protected using a Message Authentication Code (MAC). The calculation of this code is based on the existence of a shared secret authentication key (KMAC) known by the entities communicating with each other.

4.1.1.5 The I&A dialogue and the MAC calculation procedures are fully specified in the Safe Functional Module described in [Subset-037]. These procedures are based on cryptographic techniques that use secret keys (KMAC). However, the procedures do not provide any means to create, distribute or update these keys. Moreover, their effectiveness relies on the key being secret, which can only be guaranteed using secure key management functions.

4.2 KMS reference architecture

4.2.1 Architecture overview

4.2.1.1 The following figure depicts the entities involved in the Key Management System.

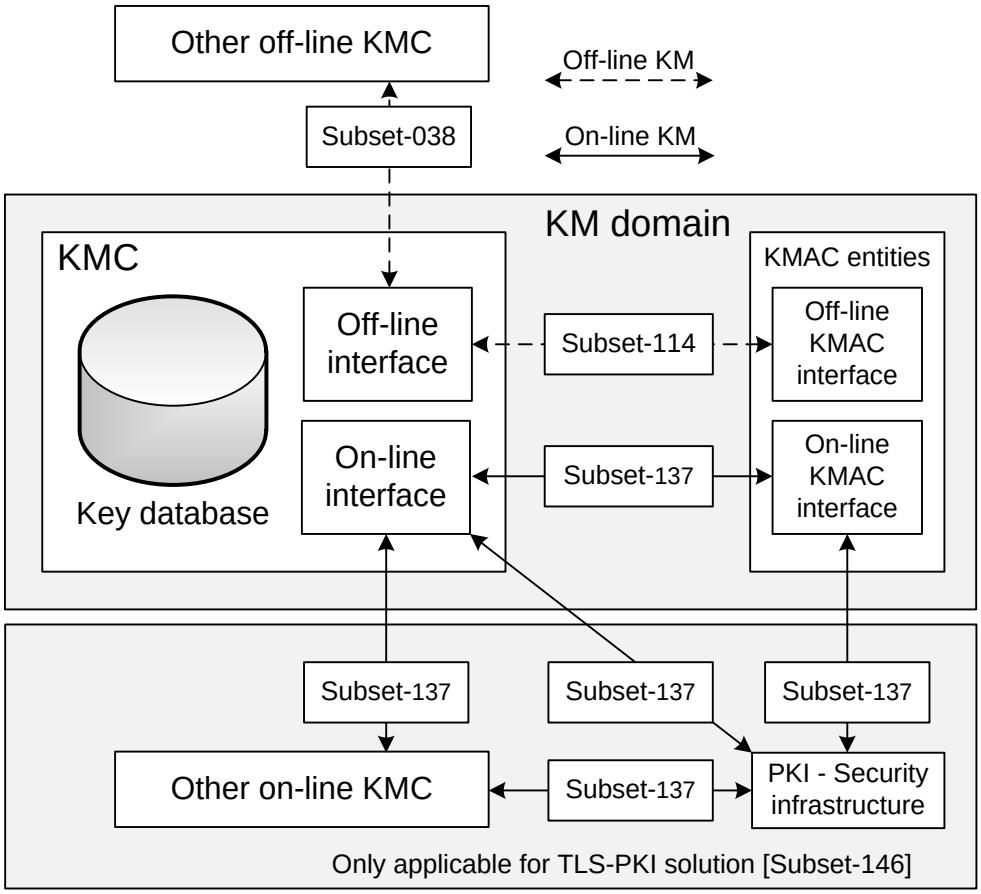

Figure 1 – KMS Reference Architecture

4.2.1.2 A KM domain is defined as one KMC and all the KMAC entities using that KMC for their key management; each KMAC entity referring to only one KMC for its key management. A KMC could administrate only trackside or on-board entities or a mix of both.

4.2.1.3 The Home KMC is the KMC that manages the key entries for a specific KMAC entity. All KMAC entities belonging to the same KM domain have the same Home KMC.

4.2.1.4 The interfaces for off-line KMS are covered in [Subset-038] and [Subset-114].

4.2.1.5 The on-line interface between KMS entities allows a KMC to manage the authentication keys (KMAC) with the KMAC entities in its domain and with other KMCs, ensuring confidentiality, integrity and authenticity.

4.2.1.6 The interface between the KMS entities and the security infrastructure allows any KMS entity to exchange digital certificate related information with the security infrastructure. The KMS entities communicate with the PKI for two main reasons:

a) to request or renew its own digital certificate;

b) to check if a given certificate issued by that PKI is (still) valid.

4.2.1.7 It's important to remark that different kinds of networks can impose some restrictions and/or performance limitations. For example, the network used between KMCs or between a KMC and a KMAC trackside entity is likely to have high speed and low latency. On the other hand, the network between a KMC and a KMAC on-board entity is likely to have lower speed and bigger latency. Furthermore, it is only the KMAC on-board entity that establishes a connection with a KMC, and an on-board entity might not be able to contact the KMC (e.g. no GPRS coverage) for some period of time.

4.2.2 KMAC

4.2.2.1 KMAC is specified in § 4.2 of [Subset-114].

4.2.2.2 Each KMAC is uniquely identified by the key serial number and the expanded ETCS-ID of the KMC that generated the key.

4.2.3 KMAC validity period

4.2.3.1 The validity period shall be defined by the beginning of validity date followed by the end of validity date of the KMAC. The validity date shall be coded in HH DD MM YY format using BCD and 24 Hours format. E.g. 15 01 01 05 would mean 1st January 2005 at 3:00 PM.

4.2.3.2 The beginning date is included in the validity period, while the end date is excluded. Examples:

- beginning date "03 01 01 05" means that the key is valid from 3 AM, the 1st January 2005;
- end date "03 01 01 05" means that the key becomes invalid at 3 AM, the 1st January 2005.

4.2.3.3 UTC time shall be used for the interface.

4.2.3.4 The specific format 0xFFFFFFFF can be used for the end date only to specify infinite validity period.

4.2.3.5 How to check the key validity period is specified in [Subset-037].

4.2.4 KMC

4.2.4.1 The KMC is responsible for the generation of the authentication keys (KMAC) needed to establish a safe connection between a KMAC trackside entity belonging to its domain and any KMAC on-board entity operating in its domain.

4.2.4.2 The KMC issuing or updating a key entry is responsible to guarantee that the validity period for this key entry does not overlap with any other validity period of any other key entry applicable to any connection to which a current key entry is applicable.

4.2.4.3 When an authentication key is needed to establish a safe connection between RBCs belonging to different KM domains, the KMC responsible for generating the key shall be agreed between the operators.

4.2.4.4 The KMC shall uniquely identify all its generated keys with a key serial number.

4.2.4.5 Even if it is possible to allocate the same KMAC value to connections related to different on-board equipment, the identifier of each authentication key related to different on-board equipment connections shall still be unique.

4.2.4.6 Even if it is possible to allocate the same KMAC value for more than one RBC-RBC connection, the identifier shall be unique for each RBC-RBC connection.

4.2.4.7 The KMC is also responsible for installing, updating, and deleting key entries (KMAC and related information) in all KMAC entities belonging to its domain.

4.2.4.8 The KMC shall be able to process requests for generation, installation, update and deletion of key entries from another KMC.

4.2.4.9 The KMC shall be able to request for generation, installation, update and deletion of key entries to another KMC.

4.2.4.10 The KMC shall report key status update to a KMC having requested generation, update, installation or deletion of key entries.

4.2.4.11 The KMC shall only request another KMC to update or delete keys which the requesting KMC has issued.

4.2.4.12 If requested by another KMC to install, update or delete keys, the KMC shall check that these keys were issued by that other KMC.

4.2.4.13 The KMC shall be able to check the key database in KMAC entities belonging to its KM domain.

4.2.4.14 It is the responsibility of the KMC to recover from any KM related degraded cases occurring in a KMAC entity. This has to be done according to the KM domain's own rules, e.g. by deleting and reinstalling all keys in this KMAC entity.

4.2.5 KMAC entity

4.2.5.1 A KMAC entity shall refer to only one Home KMC.

4.2.5.2 KMAC entities shall use only their Home KMC for key management purposes.

4.2.5.3 The KMAC entity shall not modify or delete any key entry installed by the Home KMC unless ordered to do that by the Home KMC.

4.2.5.4 The KMAC entity shall guarantee that key management transactions do not affect any already established connections for train supervision.

4.2.5.5 An updated authentication key will not be applied to an active connection. The key will take effect the next time the connection is established.

4.2.5.5.1 Note: For long-lasting connections like the RBC-RBC interface, there may be a need for an operational procedure to re-initiate the connection.

4.2.6 KMAC on-board entity

4.2.6.1 The KMAC on-board entities shall contact their Home KMC on a regular basis in order to check if any key update is needed.

4.2.6.2 The KMAC on-board entity shall contact its Home KMC if any of the following conditions is fulfilled:

a) The ERTMS/ETCS on-board equipment is switched on and the start-up tests, if any, are completed successfully.

b) The time elapsed since the last successfully completed session with the Home KMC is longer than a predefined time period configured in the on-board. This time period value is defined by the Home KMC and shall be between 1 hour and 1000 hours, with the default value being 10 hours.

c) The KMAC on-board entity maintenance staff requests a key update.

d) The KMAC on-board entity detects an invalid or corrupted KMAC key.

4.2.6.3 If the on-board entity is not able to complete successfully the connection with its Home KMC, the KMAC on-board entity shall retry to establish the session with its Home KMC every 10 minutes.

4.3 On-line interface overview

4.3.1 Security interface overview

4.3.1.1 In order to achieve confidentiality, authenticity and integrity of the distributed cryptographic material (KMAC), the TLS protocol has been chosen. See Annex A in [Subset-146].

4.3.2 Application protocol overview

4.3.2.1 The application protocol allows distribution, update and deletion of key entries between two KMCs and from KMC to KMAC entities.

4.3.2.2 The application protocol also provides means to request key operations, to perform a key database consistency check and to inform about the key distribution status.

4.3.3 Transport protocol overview

4.3.3.1 The TLS protocol is a layer on top of the TCP/IP protocol stack. Therefore, KMS entities shall be able to establish or accept TCP connections from peer entities in order to implement the on-line interfaces seen in Figure 1.

4.3.3.2 KMS entities shall also be able to establish TCP connections with the PKI because the distribution and validation of digital certificates rely on TCP/IP.

4.3.3.3 To avoid impact on the ERTMS/ETCS services, the KMS functions shall use an APN separate from the one used for ETCS operations.

4.4 Random number generation

4.4.1.1 The implementation of key generation and secure communication protocols requires the use of cryptographically secure random numbers. A cryptographically secure random or pseudo-random number generator shall be used when generating the KMAC (see §4.2.4).

4.4.1.2 The random number generator, its use and implementation, shall fulfil the requirements stated in [ENISA] § 6.2.

4.4.1.2.1 Note: In the case of using a pseudorandom number generator, special attention has to be paid to the initialisation process and to the secrecy of the pseudorandom number generator seed.

5. APPLICATION INTERFACE SPECIFICATIONS

5.1 Scope and purpose

5.1.1.1 This chapter specifies the on-line KMS application interface and consists of:

a) Functional specification

b) Message specification

c) Data flow management

5.1.1.2 In this chapter, the term “Key” refers only to the authentication key, i.e. the KMAC.

5.1.1.3 The following functions are specified for the on-line KMS interface:

a) Add Keys

b) Delete Keys

c) Delete All Keys

d) Update Key Validities

e) Update Key Entities

f) Check Key Database

g) Report Key Update Status

h) Request Key Operation

5.2 Functional specification

5.2.1 Introduction

5.2.1.1 The following sections specify the functions needed for on-line key management between two KMCs or between the KMC and KMAC entities.

5.2.1.2 Additional functions could exist locally but shall not interfere with this Subset. The KM domain administrator is responsible for common understanding of any local functions.

5.2.1.3 All functions specified in § 5.2 are mandatory.

5.2.1.4 Each function specified in § 5.2 constitutes a complete transaction, i.e. a request from an entity and the response to this request.

5.2.2 Add Keys

5.2.2.1 This function is used by the KMC either to install one or more authentication keys (KMAC) into a KMAC entity or to exchange keys with another KMC.

5.2.2.2 The function “Add Keys” shall define:

a) the authentication key (KMAC) to be installed;

b) the recipient KMAC entity;

c) the list of KMAC entities associated with this key;

d) the validity period associated with this key.

5.2.2.3 To install one or several key entries in a KMS entity, the KMC shall send an “Add Keys” command message including one request per key entry that shall be installed.

5.2.2.4 When a KMS entity receives an “Add Keys” command message that passes the header and message structure verification, it shall respond with a notification message with one answer for each key entry included in the command message. Each reply shall indicate the result of the installation of the corresponding key entry.

5.2.3 Delete Keys

5.2.3.1 This function is used by the KMC for:

a) deleting one or more key entries in a KMAC entity;

b) deleting one or more key entries in another KMC.

5.2.3.2 To delete one or several key entries in a KMS entity, the KMC shall send a “Delete Keys” command message including one request per key entry that shall be deleted.

5.2.3.3 When a KMS entity receives a “Delete Keys” command message that passes the header and message structure verification, it shall respond with a notification message with one answer for each key entry included in the command message. Each reply shall indicate the result of the deletion of the corresponding key entry.

5.2.3.4 The deletion shall be performed in such a way that the deleted keys cannot be recovered.

5.2.4 Delete All Keys

5.2.4.1 This function is used by the KMC for deletion of all key entries stored in a KMAC entity.

5.2.4.2 To delete all key entries in a KMAC entity, the KMC shall send a “Delete All Keys” command message.

5.2.4.3 When a KMAC entity receives a “Delete All Keys” command message that passes the header and message structure verification, it shall respond with a notification message indicating the result of the deletion.

5.2.4.4 The deletion shall be performed in such a way that the deleted keys cannot be recovered.

5.2.5 Update Key Validity Periods

5.2.5.1 This function is used by the KMC for:

a) updating the validity period of already distributed keys in a KMAC entity;

b) updating the validity period of already distributed keys in another KMC.

5.2.5.2 To update the validity period for one or several key entries in a KMS entity, the KMC shall send an “Update Key Validity Periods” command message including one request per key entry that shall be updated.

5.2.5.3 When a KMS entity receives an “Update Key Validity Periods” command message that passes the header and message structure verification, it shall respond with a notification message with one reply for each requested update of key validity period requested by

the command message. Each reply shall indicate the result of the update of the key validity period of the corresponding key entry.

5.2.5.4 The validity period updated by the “Update Key Validity Periods” command message shall replace the previous validity period associated with the corresponding key entry.

5.2.6 Update Key Entities

5.2.6.1 This function is used by the KMC for:

a) updating the list of KMAC entities linked to already installed keys in a KMAC entity;

b) updating the list of KMAC entities linked to already distributed keys in another KMC.

5.2.6.2 To update the list of KMAC entities for one or several key entries in a KMS entity, the KMC shall send an “Update Key Entities” command including one request per key entry that shall be updated.

5.2.6.3 When a KMS entity receives an “Update Key Entities” command message that passes the header and message structure verification, it shall respond with a notification message with one reply for each update of key entities requested by the command message. Each reply shall indicate the result of the update of the key entities of the corresponding key entry.

5.2.6.4 The list of KMAC entities updated by the “Update Key Entities” command message shall replace any previously distributed list of KMAC entities associated with the corresponding key entry.

5.2.7 Check Key Database

5.2.7.1 This function is used by the KMC for requesting the checksum computed on the key database of a KMAC entity. The returned checksum is used by the KMC to check status of the KMAC entity key database.

5.2.7.2 The key database checksum shall be calculated as stated in § 5.6.

5.2.7.3 To initiate a check of the key database status in a KMAC entity, the KMC shall send a “Request Key Database Checksum” inquiry message.

5.2.7.4 When a KMAC entity receives a “Request Key Database Checksum” message from its Home KMC, it shall calculate a checksum on its key database and respond with a notification message reporting the computed checksum.

5.2.7.5 When the KMC receives the notification message including the checksum, it uses this value to check the status of KMAC entity key database.

5.2.8 Report Key Update Status

5.2.8.1 This function is used by the KMC to report a status change of a key entry in a KMAC entity in its KM domain to the KMC that issued the key. The key status could have changed either due to a request from the KMC that issued the key or due to events in the KMAC entity’s KM domain.

5.2.8.2 When a KMC has successfully installed a key issued by another KMC, it shall report this to the issuing KMC.

5.2.8.3 When a KMC has successfully updated the validity period or the list of KMAC entities for a key issued by another KMC, it shall report this to the issuing KMC, unless there is a pending update for this key.

5.2.8.4 When a KMC has successfully deleted a key issued by another KMC, in the relevant KMAC entity and in its own key database, it shall report this to the issuing KMC. If a key was deleted without ever having been installed in a KMAC entity, the KMC shall respond after deleting the key entry from its own key database.

5.2.8.5 When a KMC receives a “Report Key Update Status” notification message from another KMC, it shall update the status of the key entry in its database and reply that the reported status of the key has been taken into account.

5.2.8.6 Management of key update degraded cases is in the scope of the KMAC entity’s Home KMC, and failure to install, delete or update a key entry in a KMAC entity is not reported to the issuing KMC.

5.2.9 Request Key Operation

5.2.9.1 This function is used by the KMC for requesting an issuing KMC to generate, update or delete key entries for a KMAC entity belonging to the requesting KM domain.

5.2.9.2 The request shall specify one of the following reasons for the key operation:

a) New train operating in the issuing KM domain;

b) Modification of the area of operation in the issuing KM domain;

c) Reduction of scheduled permission in the issuing KM domain (i.e. the date of end of operation of the KMAC entity in the issuing KM domain is set earlier than the date of end of validity of the KMAC distributed to this KMAC entity);

d) Approaching the end of validity period for some of the issued keys.

5.2.9.3 To request another KMC to perform a key operation, the KMC shall send a “Request Key Operation” message including the identity of the KMAC entity for which the key operation is requested.

5.2.9.4 When an issuing KMC receives a “Request Key Operation” command message that passes the header and message structure verification, it shall respond with a notification message indicating that the key operation request has been received and including the maximum time required for responding to the request.

5.2.9.5 The issuing KMC can respond to a request for key operation by adding, updating or deleting a key entry.

5.2.9.5.1 Note: The requesting KMC should not make any assumptions about how the issuing KMC will respond to the request for key operation. E.g.: a reduction of scheduled permission to the current date for a decommissioned train or for a train no more operating

in the issuing KM domain could be responded to with a key deletion request or with a key validity period update request.

5.2.9.6 In case the KMC is not able or allowed to perform the key operation requested within the time indicated in the response to the “Request Key Operation”, this is not reported to the requesting KMC. If this time elapses, the situation needs to be handled by some operational procedure. The definition of such operational procedures is out of scope of this document.

5.3 Message definition

5.3.1 Introduction

5.3.1.1 This section defines the structure of the messages exchanged between KMS entities in order to implement the functions listed in section 5.2.

5.3.1.2 Messages are divided into Command, Inquiry and Notification:

a) Command messages require some modification of the key database in the receiving KMS entity

b) Inquiry message requests only a response from the receiving KMS entity without any modification of the key database

c) Notification messages are used as:

- reply to a message
- notification of TLS session establishment
- notification of update status
- notification of end of update

5.3.1.3 The following table lists the Command messages:

Message - Command	Message flow direction		
CMD_ADD_KEYS	KMC	→	KMS entity
CMD_DELETE_KEYS	KMC	→	KMS entity
CMD_DELETE_ALL_KEYS	KMC	→	KMAC entity
CMD_UPDATE_KEY_VALIDITIES	KMC	→	KMS entity
CMD_UPDATE_KEY_ENTITIES	KMC	→	KMC/OBU
CMD_REQUEST_KEY_OPERATION	KMC	→	KMC

5.3.1.4 The following table lists the Inquiry message:

Message – Inquiry	Message flow direction		
INQ_REQUEST_KEY_DB_CHECKSUM	KMC	→	KMAC entity

5.3.1.5 The following table lists the Notification messages:

Message – Notification	Message flow direction		
NOTIF_KEY_UPDATE_STATUS	KMC	→	KMC
NOTIF_ACK_KEY_UPDATE_STATUS	KMC	→	KMC
NOTIF_SESSION_INIT	KMC	→	KMS entity
	KMS entity	→	KMC
NOTIF_END_OF_UPDATE	KMC	→	KMS entity
NOTIF_RESPONSE	KMS entity	→	KMS entity
NOTIF_KEY_OPERATION_REQ_RCVD	KMC	→	KMC
NOTIF_KEY_DB_CHECKSUM	KMAC entity	→	KMC

5.3.1.6 Command messages can carry several requests of the same type, but it is not possible to mix different types of requests in the same Command message.

5.3.1.7 A Notification message replying to a Command message shall include either one result per request, in the same order as the requests, in the Command message to which it replies or only the response field, indicating the failure in the execution of the Command message.

5.3.2 Format and check of messages

5.3.2.1 All messages are specified in binary format and all values are serialized in network byte order (Big Endian).

5.3.2.2 All messages consist of a message header which is optionally followed by a message body. The general message structure is depicted below:

Figure 2 – General message structure

5.3.2.3 The common message header specifies the type of information in the body (if any).

5.3.2.4 The message size shall not exceed 5000 bytes.

5.3.2.5 In the tables, the following conventions apply:

a) **Description** provides a short explanation of the message/structure.

b) **Field** provides the reference name for the information contained in the message.

c) **Size** of a field is provided in bytes.

d) **Values** shall be coded as unsigned integers.

e) **Field description** provides a short explanation of the field.

f) Range of allowed values can be specified as a closed interval from X to Y as follows: [X..Y].

g) An empty Value field means that the full range is available.

h) A repeated field is specified as F[N], which means that there are “N” occurrences of the single field “F” in the message.

5.3.2.6 When a KMS entity receives a message, it shall verify the header and message structure. If there is any error in the header or message structure it shall discard the message and respond with a notification message (see NOTIF_RESPONSE) reporting the error which has occurred (see RESPONSE field).

5.3.2.7 Verification of the message header and structure shall include the following:

a) check that the header of the message contains the unique identifier of the receiving entity (see Receiver ID field);

b) check that the header of the message contains the unique identifier of the entity authenticated for the current connection (see the Sender ID field);

c) check that the value of each field is within the allowed value range;

d) check that the Message Length field in the header corresponds to the sum of the parts of which the message consists, such that, when parsing the message, no data would be read outside the message and no data would be left unparsed at the end of the message;

e) check that the request corresponds to a supported request (see Message type field);

f) check that the header of the message contains a supported version of the interface.

5.3.2.8 For every message exchanged on the on-line KMS interface, each key shall be identified unambiguously (see K-IDENTIFIER field).

5.3.2.9 In the following tables, the term “undefined” means that the value can be used for local implementations but this may lead to compatibility issues. The term “reserved” means that the values are reserved for future use within the scope of this document.

5.3.3 Message header

Description	Message Header used in all messages.		
Field	**Size**	**Value**	**Field description**
Message Length	4	[20..5000]	Total length of this message including header and body in bytes.
Interface Version	1	2	Version of the interface. Note: only version “2” is currently available.

Receiver ID	4	ETCS-ID-EXP	The unique identifier of the intended recipient of the message.
Sender ID	4	ETCS-ID-EXP	The unique identifier of the sender of the message.
Transaction Number	4	[1..2^{32}-1]	The Transaction Number identifies a transaction with a particular set of operations to be performed. The Transaction Number of the message being responded to shall be used as Transaction Number in the response.
		0	Transaction Number to be used in messages that do not require a reply, are not a reply to a request or are a notification response reporting a transaction or sequence number mismatch: • NOTIF_SESSION_INIT • NOTIF_END_OF_UPDATE • NOTIF_RESPONSE (Transaction Number mismatch or Sequence Number mismatch)
Sequence Number	2	[0..65535]	The Sequence Number allows checking messages for sequence errors, i.e. lost or repeated messages. The sequence number shall wrap around to 0 after 65535.
Message type	1	0	CMD_ADD_KEYS
		1	CMD_DELETE_KEYS
		2	CMD_DELETE_ALL_KEYS
		3	CMD_UPDATE_KEY_VALIDITIES
		4	CMD_UPDATE_KEY_ENTITIES
		5	CMD_REQUEST_KEY_OPERATION
		6	INQ_REQUEST_KEY_DB_CHECKSUM
		7	NOTIF_KEY_UPDATE_STATUS
		8	NOTIF_ACK_KEY_UPDATE_STATUS
		9	NOTIF_SESSION_INIT
		10	NOTIF_END_OF_UPDATE
		11	NOTIF_RESPONSE
		12	NOTIF_KEY_OPERATION_REQ_RCVD
		13	NOTIF_KEY_DB_CHECKSUM
		[14..200]	Reserved
		[201..255]	Undefined

5.3.3.1 ETCS-ID-EXP consists of the following fields:

Description	The unique identifier for a KMS entity.		
Field	**Size**	**Value**	**Field description**
ETCS-ID type	1		ETCS-ID type as specified in [Subset-037]
ETCS-ID	3		Entity ETCS-ID as specified in [Subset-037]

5.3.4 CMD_ADD_KEYS

Description	Message for adding key entries to the receiver's key database.		
Field	**Size**	**Value**	**Field description**
REQ-NUM	2	[1..97]	The number of K-STRUCT structures that follow.
K-STRUCT [REQ-NUM]	*		*The size of this field depends on: • Number of key entries • Number of KMAC entities per key entry

5.3.4.1 K-STRUCT consists of the following fields:

Description	Structure to describe a key entry.		
Field	**Size**	**Value**	**Field description**
K-LENGTH	1	24	The key length in bytes (KMAC)
K-IDENTIFIER	8		Structure that uniquely identifies a key
ETCS-ID-EXP	4		The expanded ETCS-ID of the recipient KMAC entity
KMAC	K-LENGTH		The authentication key
PEER-NUM	2	[1..1000]	The number of peer entities following this field. At least one peer entity shall be specified in K-STRUCT.
ETCS-ID-EXP [PEER-NUM]	4*PEER-NUM		List of KMAC entities linked to this key.
VALID-PERIOD	8		Validity period as specified in § 4.2.3

5.3.4.2 K-IDENTIFIER consists of the following fields:

Description	Structure to uniquely identify a KMAC.		
Field	**Size**	**Value**	**Field description**
ETCS-ID-EXP	4		The identity of the KMC that issued the key.
SNUM	4		The serial number of the key.

5.3.5 CMD_DELETE_KEYS

Description	Message for deleting key entries from the key database in the receiving KMS entity.		
Field	**Size**	**Value**	**Field description**
REQ-NUM	2	[1..500]	The number of K-IDENTIFIER structures that follow.
K-IDENTIFIER [REQ-NUM]	8*REQ-NUM		List of K-IDENTIFIER

5.3.6 CMD_DELETE_ALL_KEYS

Description	Message for deleting all key entries stored in the receiving KMAC entity. This message consists only of the message header.

5.3.7 CMD_UPDATE_KEY_VALIDITIES

Description	Message for updating the validity periods of a set of key entries.		
Field	**Size**	**Value**	**Field description**
REQ-NUM	2	[1..250]	The number of K-VALIDITY structures that follow
K-VALIDITY [REQ-NUM]	16*REQ-NUM		List of K-VALIDITY structures

5.3.7.1 K-VALIDITY consists of the following fields:

Description	Structure to update the validity period of a key entry.		
Field	**Size**	**Value**	**Field description**
K-IDENTIFIER	8		Structure that uniquely identifies a key
VALID-PERIOD	8		Validity period as specified in § 4.2.3

5.3.8 CMD_UPDATE_KEY_ENTITIES

Description	Message for updating the KMAC entities of a set of key entries.		
Field	**Size**	**Value**	**Field description**
REQ-NUM	2	[1..250]	The number of K-ENTITIES structures that follow
K-ENTITIES [REQ-NUM]	REQ-NUM * (10 + 4 * PEER-NUM)		List of K-ENTITIES structures

5.3.8.1 K-ENTITIES consists of the following fields:

Description	Structure describing the KMAC entities to which a key shall be linked.		
Field	**Size**	**Value**	**Field description**
K-IDENTIFIER	8		Structure that uniquely identifies a key entry
PEER-NUM	2	[1..1000]	Number of KMAC entities following this field
ETCS-ID-EXP [PEER-NUM]	4*PEER-NUM		List of KMAC entities linked to this key

5.3.9 CMD_REQUEST_KEY_OPERATION

Description	Message for requesting the issuing KMC to perform a key operation for a KMAC entity.		
Field	**Size**	**Value**	**Field description**
ETCS-ID-EXP	4		KMAC entity for which a key operation is requested.
REASON	1	0	New train operating in the issuing KM domain
		1	Modification of the area of operation in the issuing KM domain
		2	Reduction of scheduled permission in the issuing KM domain
		3	Approaching the end of validity period for some of the issued keys
		[4..200]	Reserved
		[201..255]	Undefined

VALID-PERIOD	8		Field to be included only if REASON = 2 Validity period as specified in § 4.2.3. Beginning date of validity period shall be equal to the beginning of the validity period of the key for which a request for reduction of scheduled permission is issued. End date of validity period shall be set to the date requested for reduction of scheduled permission.
TEXT-LENGTH	2	[0..1000]	Length of the optional text
TEXT	TEXT-LENGTH		Optional text to provide some extra information for a key operation request (if TEXT_LENGTH > 0). Text is encoded using UTF-8.

5.3.10 INQ_REQUEST_KEY_DB_CHECKSUM

Description	Message for requesting a KMAC entity to compute the checksum over its key database and report the result to the KMC. This message consists only of the message header.

5.3.11 NOTIF_KEY_UPDATE_STATUS

Description	Message for reporting status for a key to the issuing KMC.		
Field	**Size**	**Value**	**Field description**
K-IDENTIFIER	8		Identifier of the key for which the status is reported.
K-STATUS	1	1	The key is installed
		2	The key is updated
		3	The key is deleted

5.3.12 NOTIF_ACK_KEY_UPDATE_STATUS

Description	Message for acknowledging the reception of a NOTIF_KEY_UPDATE_STATUS message for a specific key. This message consists only of the message header. Note: Message is send in case of positive acknowledgement

5.3.13 NOTIF_SESSION_INIT

Description	Message for initialising a new session. This message informs the peer entity about the initial sequence number, the list of supported interface versions and the application time-out value. The sequence number in the header shall be used as the initial sequence number. The header of this message shall always conform to version “2” for backward compatibility.		
Field	**Size**	**Value**	**Field description**
N-VERSION	1	1	Number of versions of the interface supported by the entity. Only one version is supported in the current release.
INTERFACE-VERSION [N-VERSION]	N-VERSION	2	List of supported versions. Only version “2” of the on-line interface shall be supported by all entities on the current release of the interface.
APP-TIME-OUT	1	[5..254]	Application time-out in seconds.
		255	Application time-out defined by the peer entity.

5.3.14 NOTIF_END_OF_UPDATE

Description	Message for indicating that all requested updates have been transferred. It is sent after all updates have been acknowledged and no further command has to be sent to the KMS entity. This message consists only of the message header.

5.3.15 NOTIF_RESPONSE

Description	Message for reporting the result of an Inquiry or Command message to the originator of that message. The first field indicates the result or an error of some kind, optionally followed by an individual result for each request.		
Field	**Size**	**Value**	**Field description**
RESPONSE	1	0	If the Command message responded to contains a list of requests (i.e. contains “REQ-NUM” field), “0” means that the message verification was successful. Confirmation of each request follows in the list of NOTIFICATION_STRUCT. If the response is to an Inquiry message or to a Command message that does not contain a list of requests, “0” means that the message verification was successful and the request has been successfully processed.
		1	Request not supported (see § 5.3.2.7 e).
		2	Message length error (see § 5.3.2.7 d).
		3	Sender ID included in the request doesn’t match the ETCS-ID-EXP of the expected peer KMS entity (see § 5.3.2.7 b).
		4	Receiver ID included in the request doesn’t match the KMS entity’s ETCS-ID-EXP (see § 5.3.2.7 a).
		5	Unsupported interface version (see § 5.3.2.7 f).
		6	Unrecoverable key database. This value is used by the KMAC entity to report the need for a complete key database reinstallation (e.g. after detecting an invalid or corrupted KMAC)..
		7	Failure in processing the request. This value shall only be used for reporting errors in the processing of messages that do not include a list of requests: CMD_DELETE_ALL_KEYS; CMD_REQUEST_KEY_OPERATION; INQ_CHECK_KEY_DB.
		8	Checksum mismatch (see § 5.2.7.4).
		9	Sequence number mismatch (see § 5.4.4.4).
		10	Transaction number mismatch (see § 5.4.4.5).
		11	Format error (see § 5.3.2.7 c).
		[12..254]	Reserved.

		255	Other error.
REQ-NUM	2	[0..500]	The number of NOTIFICATION_STRUCT that follows. This field shall be “0” if • the RESPONSE field value is different from “0”. • the response is to an Inquiry Message • the response is to a Command Message that did not contain a list of requests.
NOTIFICATION_STRUCT [REQ-NUM]	REQ-NUM		List of NOTIFICATION_STRUCT structures

5.3.15.1 NOTIFICATION_STRUCT consists of the following fields:

Description	The result of a single command for a key entry.		
Field	**Size**	**Value**	**Field description**
RESULT	1	0	Request successfully processed
		1	Unknown key: key not found in the KMS entity database
		2	Maximum number of keys exceeded in the KMS entity database
		3	Request to install a key already installed in the KMAC entity database. The installation request will not be processed
		4	Key corrupted
		5	Recipient expanded ETCS-ID mismatch
		[6..254]	Reserved
		255	Other error

5.3.16 NOTIF_KEY_OPERATION_REQ_RCVD

Description	Message for reporting that the command CMD_REQUEST_KEY_OPERATION has been received. This message also indicates the maximum time required to respond to the key operation request.		
Field	**Size**	**Value**	**Field description**
MAXTIME	2		Maximum time (in hours) required to respond to the key operation request

5.3.17 NOTIF_KEY_DB_CHECKSUM

Description	Message for reporting the KMAC entity checksum value.		
Field	**Size**	**Value**	**Field description**
CHECKSUM	20		The checksum of the KMAC entity's key database Note: Checksum is a 16 Bytes value. 4 first bytes being set to "0".

5.4 Data flow management

5.4.1 Connection establishment

5.4.1.1 The KMC is responsible for establishing the connection with KMAC trackside entities.

5.4.1.2 The KMAC on-board entity is responsible for establishing the connection with the KMC.

5.4.1.3 The KMC requesting a key generation, installation, deletion or update, or reporting a key status change is responsible for establishing the connection with the peer KMC.

5.4.1.4 Connection between KMS entities shall be established only to send Inquiry, Command or key update status Notification messages.

5.4.1.5 As soon as a TLS connection has been established between two KMS entities, both entities shall send a NOTIF_SESSION_INIT message to the peer entity. The connection is considered as established at application level at the reception of the NOTIF_SESSION_INIT message from the peer entity.

5.4.1.6 The NOTIF_SESSION_INIT message shall include the initial sequence number used for sequence management, the list of supported interface versions and the application time-out value. This message shall always use the header compliant with the version "2" of the interface.

5.4.1.7 The highest interface version supported by both entities shall then be used during the rest of the session. The "Interface Version" in the header of the following messages shall be set to the agreed interface version.

5.4.1.8 The KMS entity shall not send any other message than NOTIF_SESSION_INIT until it has received a NOTIF_SESSION_INIT message from the other KMS entity.

5.4.1.9 After having exchanged the NOTIF_INIT message between both entities, if no common version of the interface is supported, both entities shall release the TLS connection.

5.4.1.10 The application time-out value shall be defined and distributed by the KMC initiating the connection for the KMC-KMC connection and by the KMC in case of a KMC-KMAC entity connection. The other entity shall send the specific application time-out value "Application time-out defined by the peer entity".

5.4.1.11 Once the connection is established at application level, each entity shall start to supervise the application time-out. The timer is restarted at each reception of an application message from the peer KMS entity.

5.4.1.12 NOTIF_SESSION_INIT shall not be repeated.

5.4.2 Data transmission

5.4.2.1 Once the connection between a KMC and a KMAC entity has been established, the KMC shall only send Command, Inquiry or end of update Notification messages to the KMAC entity.

5.4.2.2 In a KMC-KMC connection, only the KMC having established the connection shall request a key generation, installation, deletion or update, or report a key status change.

5.4.2.3 After sending a message for which a reply is expected, the KMC shall not send any other message until it has received a reply with the same Transaction Number as in the message it sent.

5.4.2.4 The KMS entity replying to a received message, identified by a Transaction Number, shall use the same Transaction Number as in the message it replies to.

5.4.2.5 The Transaction Number in two consecutive transactions shall be different.

5.4.2.6 The KMS entities shall send messages in sequence and increment the Sequence Number by one each time a new message is sent.

5.4.2.6.1 Note: The Sequence Number may start at any valid value and does not have to be reset between sessions.

5.4.3 Connection release

5.4.3.1 Once the KMC considers all transactions completed, the KMC shall send a NOTIF_END_OF_UPDATE message and release the connection.

5.4.3.2 In KMC-KMC connections, the KMC requesting or reporting a key update or requesting key operation is responsible for releasing the connection.

5.4.3.3 If the connection between a KMAC entity and a KMC is released before the KMC has issued the NOTIF_END_OF_UPDATE message, any transaction that has not been acknowledged before a session is terminated may not have been executed.

5.4.3.4 When the connection is re-established with the KMAC entity, the KMC can check the status of the KMAC DB by sending an INQ_CHECK_KEY_DB message and by using the returned checksum to check whether a not acknowledged, transaction has been processed or not.

5.4.4 Error management

5.4.4.1 If the NOTIF_SESSION_INIT message has not been received within 15 seconds after the TLS connection has been established between two KMS entities, the TLS connection shall be released by the KMS entity detecting the connection time-out.

5.4.4.2 If no application message is received within the application time-out, the TLS connection shall be released by the KMS entity. The application time-out recommended value is 120 seconds.
Note: the recommended application time-out value is defined to be long enough to allow application message exchange on GPRS with potentially several trials..

5.4.4.3 At message reception, the KMS entity shall check the Sequence Number before the Transaction Number.

5.4.4.4 If the sequence number of a received message is not consecutive to the previous one received, the KMS entity that detects this shall send NOTIF_RESPONSE message reporting Sequence Number mismatch and then release the connection.

5.4.4.5 The KMS entity shall check the Transaction Number (§ 5.3.3 value <> "0") in messages received as reply to a message it sent. If this Transaction Number does not match the number in the message it sent, then the KMS entity shall send a NOTIF_RESPONSE message reporting Transaction Number mismatch and then release the connection.

5.5 Application message scenarios

5.5.1 Introduction

5.5.1.1 The scenarios illustrate some of the common use cases, but are only informative.

5.5.1.2 In the scenarios, the following abbreviations are used for transmitted messages:

CMD (the type of command is given by the scenario)	CMD_ADD_KEYS CMD_DELETE_KEYS CMD_DELETE_ALL_KEYS CMD_UPDATE_KEY_VALIDITIES CMD_UPDATE_KEY_ENTITIES CMD_REQUEST_KEY_OPERATION
INQ_DB_CHK	INQ_REQUEST_KEY_DB_CHECKSUM
NOTIF_INIT	NOTIF_SESSION_INIT
NOTIF_END	NOTIF_END_OF_UPDATE
NOTIF_RESP	NOTIF_RESPONSE
NOTIF_STATUS	NOTIF_KEY_UPDATE_STATUS
NOTIF_ACK	NOTIF_ACK_KEY_UPDATE_STATUS
NOTIF_REQ_RCVD	NOTIF_KEY_OPERATION_REQ_RCVD
NOTIF_CHECK	NOTIF_KEY_DB_CHECKSUM
SNex	Sequence Number x in entity e
TNx	Transaction Number x
[N]	List of N entries

5.5.1.3 A ‘box’ on the time-line means some activity taking an undefined amount of time.

5.5.1.3.1 Note: When a command is not processed, this is clearly stated in the scenario.

5.5.2 KMC–KMAC entity key management scenario

5.5.2.1 The following figure describes how to add, delete or update authentication keys in a KMAC entity.

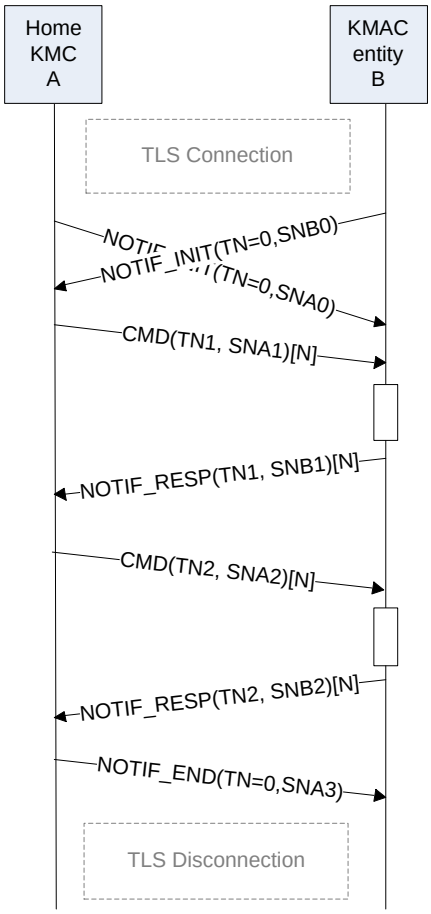

Figure 3 – KMC-KMAC entity key management scenario

5.5.2.2 As soon as the TLS connection is established, both entities send a NOTIF_SESSION_INIT message with their initial sequence number.

5.5.2.3 After receiving the NOTIF_SESSION_INIT message, the KMC sends a command message. The KMC does not send any new message until it has received the corresponding NOTIF_RESPONSE for the previous one.

5.5.2.4 The KMAC entity processes the command and replies with a NOTIF_RESPONSE using the same Transaction Number as in the command message.

5.5.2.5 Once all transactions are finished, the KMC sends a NOTIF_END_OF_UPDATE and releases the connection.

5.5.3 KMC–KMAC entity: abnormal session release

5.5.3.1 The following figure describes the scenario where a KMAC entity aborts a session.

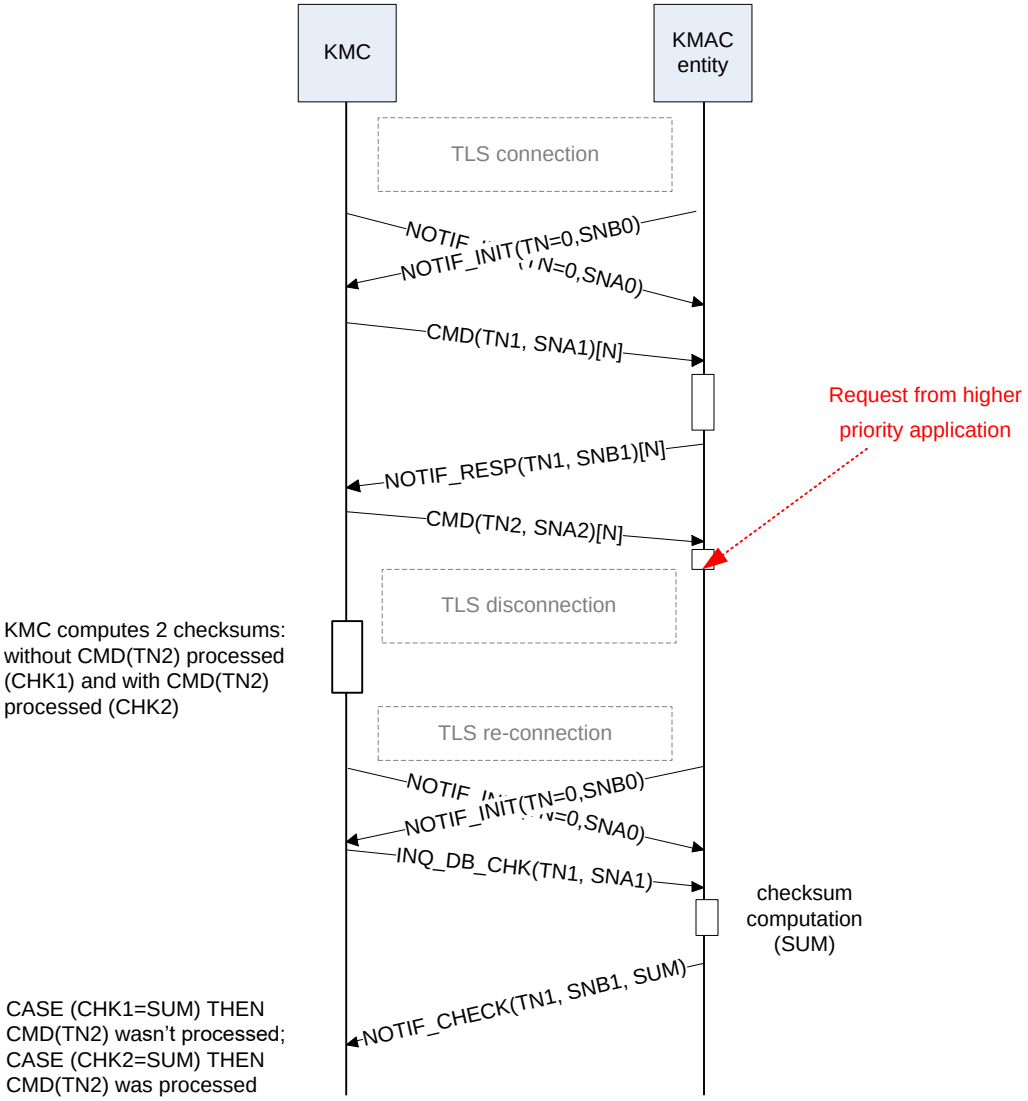

Figure 4 – KMC-KMAC entity: abnormal session release

5.5.3.2 As soon as the TLS connection is established, both entities send a NOTIF_SESSION_INIT message with their initial sequence number.

5.5.3.3 After receiving the NOTIF_SESSION_INIT message, the KMC sends a command message. The KMC does not send any new message until it has received the corresponding NOTIF_RESPONSE for the previous one.

5.5.3.4 The KMAC on-board entity processes the command and replies with a NOTIF_RESPONSE using the same Transaction Number as in the command message.

5.5.3.5 After handling the first transaction the KMAC on-board entity needs to abort the session. It releases the connection.

5.5.3.6 The KMC can determine based on the messages from the KMAC on-board entity that the first command has been executed, but the second was not. When a session is re-established with the KMAC entity, the KMC can resume the update.

5.5.4 KMC–KMC key management scenario

5.5.4.1 The following figure describes how to add, delete or update authentication keys of a KMAC entity belonging to another KM domain.

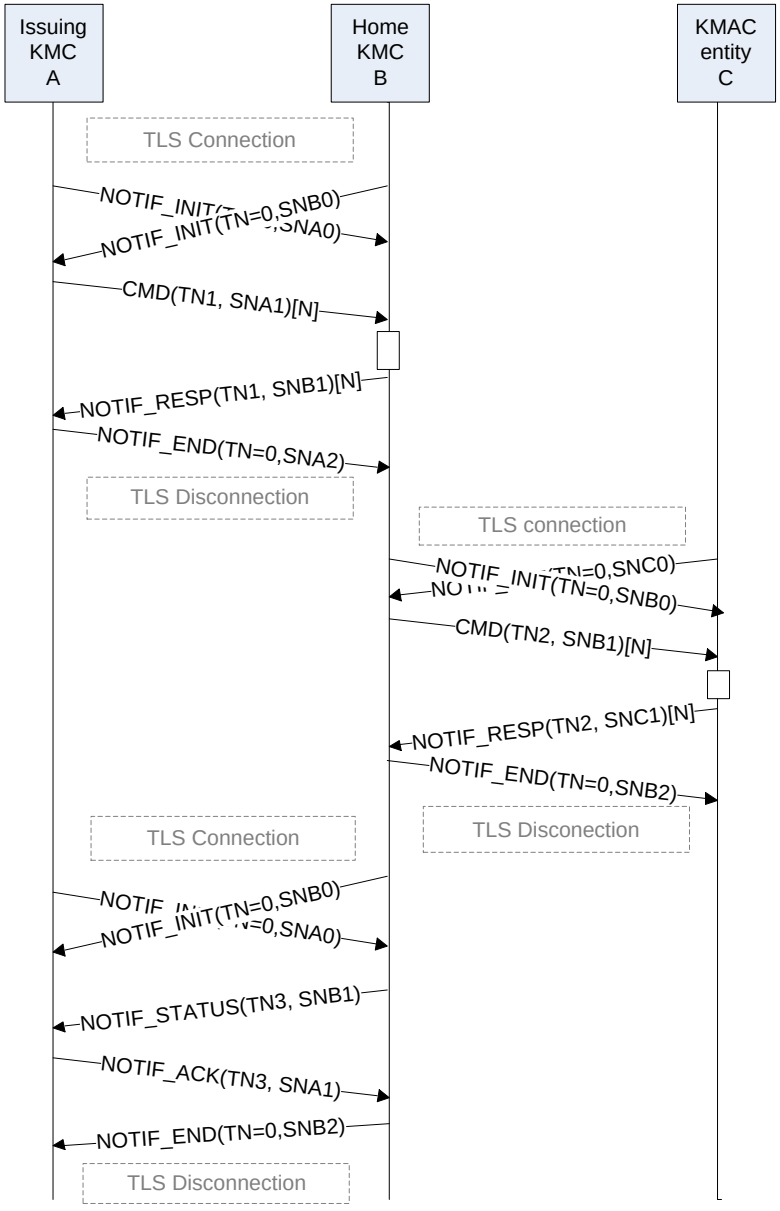

Figure 5 – KMC-KMC key management scenario

5.5.4.2 As soon as the TLS connection between the KMCs is established, both KMCs send a NOTIF_SESSION_INIT message with their initial sequence number.

5.5.4.3 After receiving the NOTIF_SESSION_INIT message, the issuing KMC sends a command message.

5.5.4.4 The KMAC entity's Home KMC processes the command and replies with a NOTIF_RESPONSE using the same Transaction Number as in the command message.

5.5.4.5 Once all transactions are finished, the issuing KMC sends a NOTIF_END_OF_UPDATE message and releases the connection.

5.5.4.6 When the KMAC entity's Home KMC and the KMAC entity whose key database shall be updated are connected, the Home KMC sends the appropriate commands to update the KMAC entity's key database.

5.5.4.7 After the Home KMC has received the NOTIF_RESPONSE for these commands, it releases the connection with the KMAC entity and establishes a new TLS connection with the issuing KMC. A new connection must be established since in the previous connection, the Home KMC was the receiver.

5.5.4.8 Once the connection between the KMCs is established, the Home KMC sends a NOTIF_KEY_UPDATE_STATUS message to the issuing KMC.

5.5.4.9 The issuing KMC acknowledges receiving the notification message with a NOTIF_ACK_KEY_UPDATE_STATUS message.

5.5.4.10 After receiving the acknowledgement, the Home KMC sends a NOTIF_END_OF_UPDATE message and releases the connection.

5.5.5 Time-out supervision scenarios

5.5.5.1 The following figures describe time-out supervision during connection establishment and during data transmission. Entity A has initiated the connection.

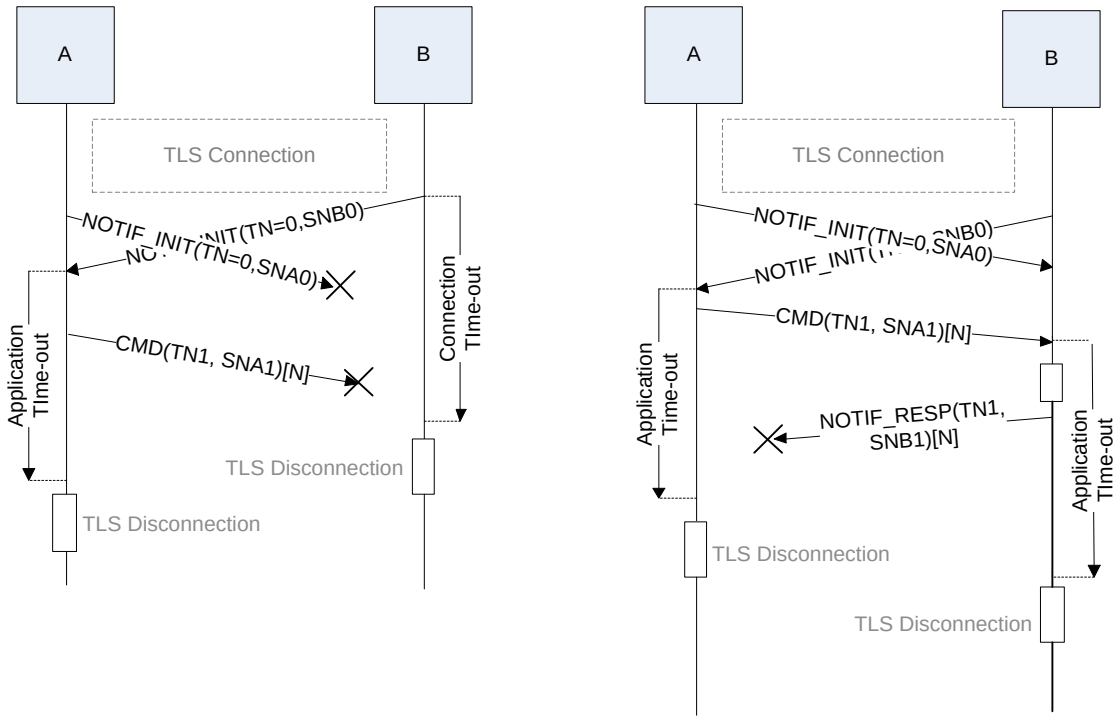

Figure 6 – Time-out supervision scenarios

5.5.5.2 As soon as the TLS connection is established, both entities send a NOTIF_SESSION_INIT message to the other entity.

5.5.5.3 Once the TLS connection is established, both entities supervise the time between receptions and checks the sequence and transaction numbers.

5.5.5.4 In the left-hand figure above, the NOTIF_SESSION_INIT message from A is lost. When the connection time-out in B expires, B releases the TLS connection. A releases the connection when the application time-out has expired.

5.5.5.5 In the right-hand figure, when the application time-out expires, both release the connection. Note that there is no repetition of KMS messages.

5.5.6 Sequence and transaction error scenarios

5.5.6.1 The following figures show sequence and transaction errors during connection establishment and during data transmission. Entity A has initiated the connection.

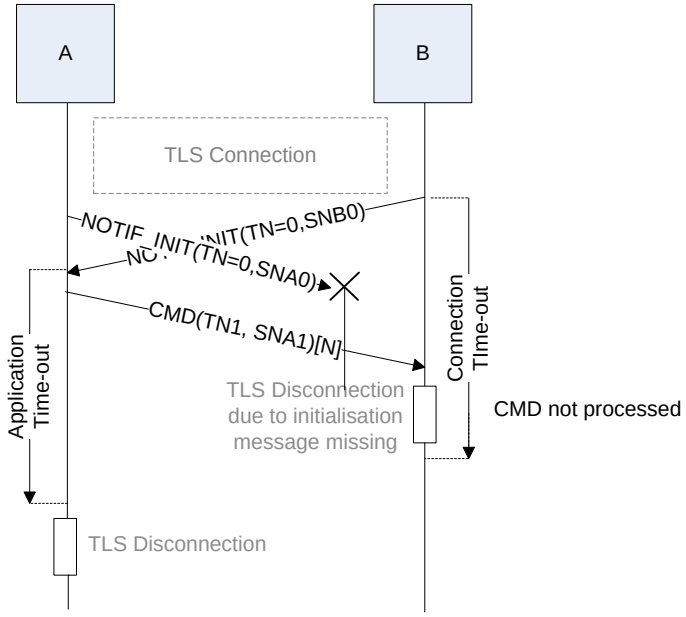

Figure 7 – Sequence error during connection establishment

5.5.6.2 As soon as the TLS connection is established, both entities send a NOTIF_SESSION_INIT message to the other entity with their initial sequence number.

5.5.6.3 Once the TLS connection is established, both entities supervise the sequence number and the transaction number, as well as the time between received messages.

5.5.6.4 In the figure above, the NOTIF_SESSION_INIT message from A is lost. When A sends a command message, B detects that a NOTIF_SESSION_INIT has not been received before receiving the command message and releases the TLS connection. A could release the connection due to the expiration of the application time-out or due to the detection of the TLS disconnection from B.

5.5.6.5 If A does not send any message before the connection time-out elapses, the connection will be released due to connection time-out.

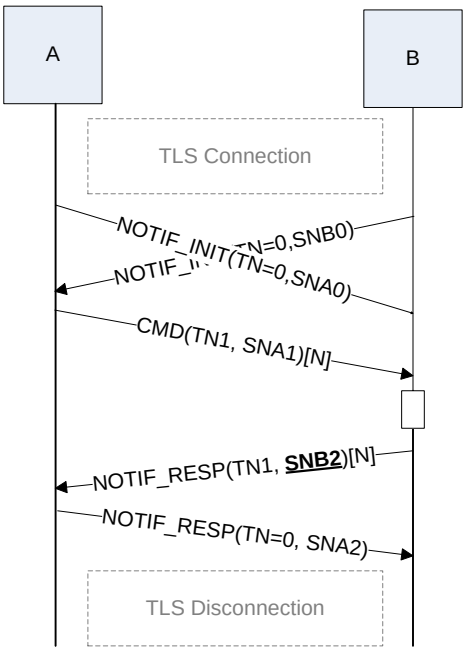

Figure 8 – Sequence number error scenario

5.5.6.6 In the figure above, when A receives a message with the wrong Sequence Number, A sends NOTIF_RESPONSE message reporting Sequence Number mismatch and releases the connection.

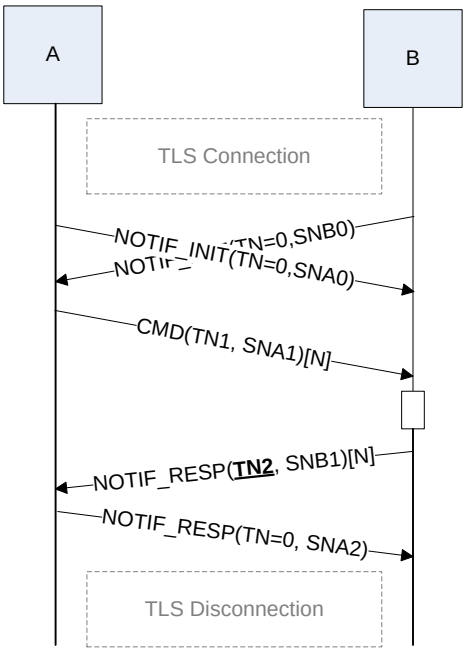

Figure 9 – Transaction number error scenario

5.5.6.7 In the figure above, when A receives a message with the wrong Transaction Number, A sends NOTIF_RESPONSE message reporting Transaction Number mismatch and releases the connection.

5.6 Definition of the Key Database checksum algorithm

5.6.1 Algorithm properties

5.6.1.1 A checksum algorithm is used to check the consistency of the key database between the Home KMC and a KMAC entity.

5.6.1.2 An overview of the checksum algorithm is illustrated in the following figure:

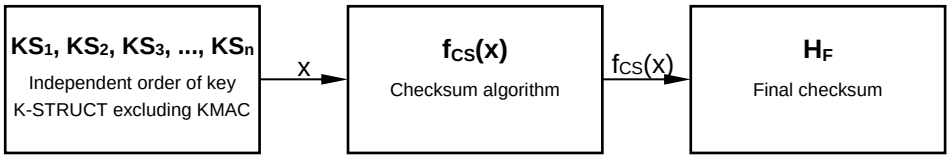

Figure 10 – Overview of the Key DB checksum algorithm

where x is the input for the checksum algorithm.

5.6.1.3 The main features of the checksum algorithm $f_{cs}(x)$ are:

- Detection of differences between key entries in the Home KMC and a KMAC entity excluding the KMAC.
- Producing the same final checksum H_F independently of the order of the input key structures KS,

$$f_{cs}(P_a(KS_1, KS_2, KS_3, ... KS_n)) = f_{cs}(P_b(KS_1, KS_2, KS_3, ... KS_n))$$

where P_a and P_b denotes different random permutations of the same key structures.

5.6.1.4 The checksum algorithm is depicted in the figure below:

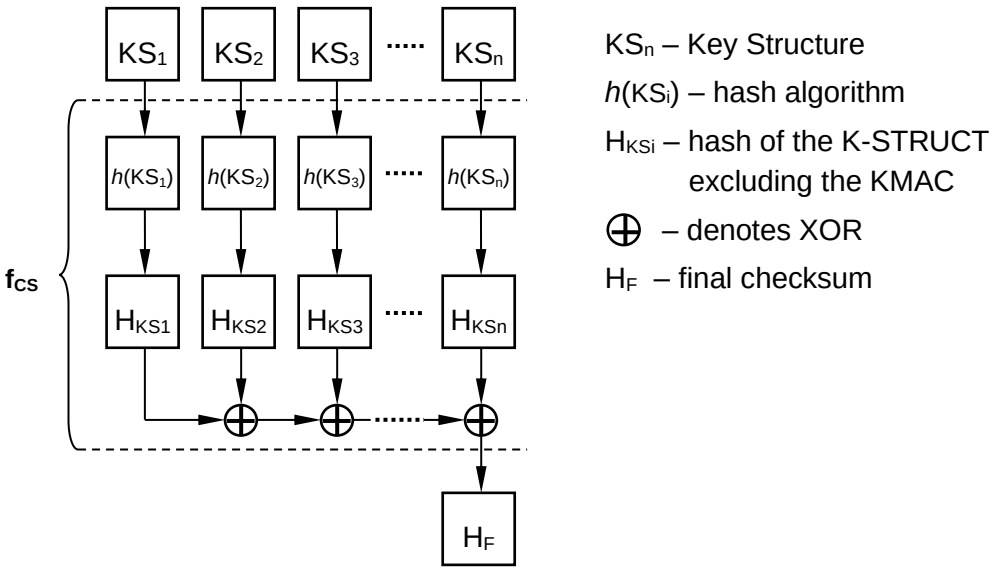

Figure 11 – Definition of the checksum algorithm

5.6.1.5 The algorithm used for the hash is MD4; for details see [RFC-1320].

5.6.1.6 Input for the hash algorithm $h(KS_i)$ consists of the fields described in the following table:

Field	Size (Bytes)	Description
K-LENGTH	1	Length of the KMAC
K-IDENTIFIER	8	Structure that uniquely identifies the KMAC
PEER-NUM	2	Number of KMAC entities that is listed following this field
ETCS-ID-EXP [PEER-NUM]	4 * PEER-NUM	List of KMAC entities linked to this key
VALID-PERIOD	8	Start and end of validity for the KMAC

Table 1: K-STRUCT excluding KMAC and ETCS-ID-EXP

5.6.1.7 An example of key database checksum computation is found in Annex A.

5.6.1.7.1 Note: the KMAC value is not used for computing the key database checksum for the following reasons:

a) KMAC corruption is very unlikely due to internal implementation checks;

b) Using the KMAC's value for the computation of the checksum will significantly reduce the strength of the KMAC as this checksum could be used to compute its value.

5.6.1.8 In case of empty key database, the checksum value shall be set to "0".

6. SECURITY INTERFACE SPECIFICATIONS

6.1.1.1 KMC, KMAC trackside entity and KMAC on-board entity shall use security Interface as defined by [Subset-146] Security Layers for ETCS Applications

6.1.1.2 KMC, KMAC trackside entity and KMAC on-board entity shall provide TLS roles as defined by [Subset-146] Annex A

7. TRANSPORT INTERFACE SPECIFICATION

7.1 Scope and purpose

7.1.1.1 This chapter specifies the information needed to establish end-to-end connections at the transport level from the on-board KMAC entity.

7.1.1.2 This involves:

a) specification of addressing;

b) definition of the TCP parameters;

c) definition of the functional interface with the EuroRadio Co-ordinating function for the KMAC on-board entities that provides the IP access service.

7.2 Chapter left intentionally free

7.3 TCP specification

7.3.1.1 For KMAC on-board entity, the TCP configuration specified in § 8.3 of [Subset-037] shall be used unless otherwise stated in this section.

7.3.1.2 The listening TCP port for the KMS application is 7912.

7.3.1.3 The recommended value for the “TcpUserTimeout” is 40 seconds.

7.3.1.4 The recommended “Max TCP segment size” for the KMS application is 550 bytes.

7.3.1.5 The values of some TCP Parameters can be proposed in the DNS TXT field, see § 8.4.1 of [Subset-037], but the applicability of such proposed values is optional, depending on the implementation.

7.4 Functional interface with EuroRadio Co-ordinating function

7.4.1.1 The KMS application uses the primitive Rm-SERVICE.request with the application type set to “KMS” to request the allocation of an IP service (see § 8.5 of [Subset-037]).

7.4.1.2 The primitive Rm-SERVICE.indication reports the result of the Rm-SERVICE.request to the KMS application (see § 8.5 of [Subset-037]), stating the service ID assigned to the KMS application and the outcome of the request through the parameters Reason and Sub-reason.

7.4.1.3 The primitive Rm-SERVICE.release is used by the KMS application to release the used IP service or by the co-ordinating function to report the release of the IP service for any reason (see § 8.5 of [Subset-037]).

ANNEX A. KEY DATABASE CHECKSUM COMPUTATION

This annex gives examples of how to compute the key database checksum.

Consider the following example (differences between each key structure marked yellow):

Example 1 is to illustrate the algorithm, serving as a complement to the definition and to give an example input, outcome pair, which allows to verify an implementation (multiple peers). **$KS_{EXAMPLE\_1}$**	**$KS_{EXAMPLE\_2}$**	**$KS_{EXAMPLE\_3}$**
K-LENGTH = 0x18 **ETCS-ID-EXP** = 0x04030201 **SNUM** = 0x0000FEDC **PEER-NUM** = 0x03 **ETCS-ID-EXP** **[1]** = 0x0100000A **ETCS-ID-EXP** **[2]** = 0x0100000B **ETCS-ID-EXP** **[3]** = 0x0100000C **VALID-PERIOD** = From 2015-03-21 14h, To 2015-03-25 18h	**K-LENGTH** = 0x18 **ETCS-ID-EXP** = 0x04030201 **SNUM** = 0x0000FEDD **PEER-NUM** = 0x03 **ETCS-ID-EXP** **[1]** = 0x0100001A **ETCS-ID-EXP** **[2]** = 0x0100001B **ETCS-ID-EXP** **[3]** = 0x0100001C **VALID-PERIOD** = From 2015-03-21 14h, To 2015-03-25 18h	**K-LENGTH** = 0x18 **ETCS-ID-EXP** = 0x04030201 **SNUM** = 0x0000FEDE **PEER-NUM**= 0x03 **ETCS-ID-EXP** **[1]** = 0x0100002A **ETCS-ID-EXP** **[2]** = 0x0100002B **ETCS-ID-EXP** **[3]** = 0x0100002C **VALID-PERIOD** = From 2015-03-21 14h, To 2015-03-25 18h

All values shall be encoded in big endian format.

Memory map of $KS_{EXAMPLE\_1}$:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	18	04	03	02	01	00	00	FE	DC	00	03	01	00	00	0A	01
16	00	00	0B	01	00	00	0C	14	21	03	15	18	25	03	15	

Resulting MD4 hash: $h(KS_{EXAMPLE\_1})$ = 0x **9D 16 B2 0B F4 25 99 E0 F8 B7 77 0A 0D DE 57 9F**

Memory map of $KS_{EXAMPLE\_2}$:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	18	04	03	02	01	00	00	FE	DD	00	03	01	00	00	1A	01
16	00	00	1B	01	00	00	1C	14	21	03	15	18	25	03	15	

Resulting MD4 hash: $h(KS_{EXAMPLE\_2})$ = 0x **75 6B 7E 1F DF 74 5D 96 32 7C 1D 4E 84 6D E8 FB**

Memory map of $KS_{EXAMPLE\_3}$:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

0	18	04	03	02	01	00	00	FE	DE	00	03	01	00	00	2A	01
16	00	00	2B	01	00	00	2C	14	21	03	15	18	25	03	15	

Resulting MD4 hash: $h(KS_{EXAMPLE\_2})$ = 0x **F3 3D 86 FB 93 A7 C7 B3 F8 90 71 CC 3E FF 39 20**

Final checksum H_F:

$$H_F = h(KS_{EXAMPLE\_1}) \oplus h(KS_{EXAMPLE\_2}) \oplus h(KS_{EXAMPLE\_3})$$

H_F = 0x **1B 40 4A EF B8 F6 03 C5 32 5B 1B 88 B7 4C 86 44**

Example 2 is to illustrate the algorithm, serving as a complement to the definition and to give an example input, outcome pair, which allows to verify an implementation (single peer).

$KS_{EXAMPLE\_1}$	$KS_{EXAMPLE\_2}$	$KS_{EXAMPLE\_3}$
K-LENGTH = 0x18	**K-LENGTH** = 0x18	**K-LENGTH** = 0x18
ETCS-ID-EXP = 0x05030201	**ETCS-ID-EXP** = 0x05030201	**ETCS-ID-EXP** = 0x05030201
SNUM = 0x0000FEDC	**SNUM** = 0x0000FEDD	**SNUM** = 0x0000FEDE
PEER-NUM = 0x01	**PEER-NUM** = 0x01	**PEER-NUM**= 0x01
ETCS-ID-EXP **[1]**	**ETCS-ID-EXP** **[1]**	**ETCS-ID-EXP** **[1]**
= 0x0200000A	= 0x0200001A	= 0x0200002A
VALID-PERIOD	**VALID-PERIOD**	**VALID-PERIOD**
= From 2015-03-21 14h,	= From 2016-03-27 14h,	= From 2017-03-28 14h,
To 2016-03-25 18h	To 2017-03-28 18h	To 2018-03-29 18h

All values shall be encoded in big endian format.

Memory map of $KS_{EXAMPLE\_1}$:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	18	05	03	02	01	00	00	FE	DC	00	01	02	00	00	0A	14
16	21	03	15	18	25	03	16									

Resulting MD4 hash: $h(KS_{EXAMPLE\_1})$ = 0x **89 05 17 41 38 40 AD AF 0B AC 98 07 68 E8 B6 6C**

Memory map of $KS_{EXAMPLE\_2}$:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	18	05	03	02	01	00	00	FE	DD	00	01	02	00	00	1A	14
16	27	03	16	18	28	03	17									

Resulting MD4 hash: $h(KS_{EXAMPLE\_2})$ = 0x **B1 3E 04 20 54 82 39 0A 56 A7 71 D5 F9 AC 67 FC**

Memory map of $KS_{EXAMPLE\_3}$:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	18	05	03	02	01	00	00	FE	DE	00	01	02	00	00	2A	14
16	28	03	17	18	29	03	18									

Resulting MD4 hash: $h(KS_{EXAMPLE\_2})$ = 0x **AD 60 00 7D BC 0C 8E 27 DC 2A EB 65 A2 DB C6 55**

Final checksum H_F:

$$H_F = h(KS_{EXAMPLE\_1}) \oplus h(KS_{EXAMPLE\_2}) \oplus h(KS_{EXAMPLE\_3})$$

H_F = 0x **95 5B 13 1C D0 CE 1A 82 81 21 02 B7 33 9F 17 C5**