ERTMS/ETCS
On-line Key Management FFFIS
REF : SUBSET-137 ISSUE : 1.0.0 DATE : 17-12-2015

Company	Technical Approval	Management approval
ALSTOM		
ANSALDO		
AZD		
BOMBARDIER		
CAF		
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
0.1.1 05-08-2015	§ 3.3, 3.4, 4.1.1.2, 4.2.3.3, Table 19, 6.3.3.5 § 3.1, 3.2, 4.1 § 3.1.1.4 § 3.3 § 3.4 § 3.4.1.1 § 4.1.1.3 § 4.2.3 § 4.3.1.6 § 3.1.1.2, 4.2.3.7, 4.3.1, 5.3.1.2, 5.3.2.2 § 5.3.3.5 § 5.3.5.4 § 5.4.2.2 § 5.6.1, 5.3.8.5 Fig. 10 § 5.6.1.6 Fig. 1 § 5.2.3 § 5.4.5, 5.2.12 § 5.2.3, 5.2.2, 5.2.4.	Review comments from EUG : • Comment 3 ; • Comment 5 ; • Comment 8 ; • Comment 11 ; • Comment 12 ; • Comment 14 ; • Comment 15 ; • Comment 21 ; • Comment 23 ; • Comment 36 ; • Comment 41 ; • Comment 42 ; • Comment 46 ; • Comment 50 ; • Comment 52 ; • Comment 54 ; • Comment 16 ; • Comment 31 ; • Comment 47 ; • Comment 30 (1)	FH
0.1.2 06-08-2015	§ 5.2, 5.3	Review comment : • Comment 27 (no revision mark for section switch) ;	FH

0.1.3 17-08-2015	§ 4.3.1, 4.3.3, 6.1, 6.2.3	Comment 2	TN
0.1.4 17-08-2015	§ 3.4 § 5.3.1.5 § 5.3.3 § 5.3.17 § 5.4.4.3 § 5.5.3	Review comment: • Comment 17 • Early disconnection function.	MA
0.1.5 17-08-2015	§ 5.1.1.3 § 5.2.10 § 5.3.1.3 § 5.3.1.5 § 5.3.3 § 5.3.9 § 5.3.16	U action key gen request	TN
0.1.6 18-08-2015	§ 5.3.3 § 5.3.15 § 5.4.5.3 § 5.4.5.4 § 5.5.6.6	Comment 48	MP
0.1.7 19-08-2015	§ 5.2.2, 5.3.3, 5.3.15 § 3.1.1.1, 4.1.1.2, 4.2.3, 5.2.9.5 § 3.3, 3.4, 4.1.1.2	Editorial update Implementation of RBC-RBC key distribution Deletion of ATO references	LA FH FH
0.1.8 25-08-2015	Fig.1, § 3.2, 4.3.1, 5.4.4, 5.3.17, 5.3.1.5, 5.3.3, 5.4.5, 5.2, 5.3.15, 4.1, 5.2.10, 5.4.1, 5.3 (for PEER-NUM and REQ-NUM)	Consolidation review in WG meeting	WG

0.1.9 15-09-2015	As marked	Internal review by SG	SP
0.1.10 21-09-2015	§ 5.2.2.2, 5.3.4.1, 5.3.15.1 § 4.2.3 See rev. marks	Update CMD_ADD_KEY after EUG remark Include validity period definition. Final editorial review in WG meeting	WG
0.2.0 22-09-2015		Release for second review	WG
0.3.0 28-10-2015	§ 3.1.1.2/3, 3.2, 3.4.1.1, 4.2.1, 3.3, 3.4, 4.2.3.5, 4.2.5.5, 4.2.6, 5.2.4.3, 5.2.9, 5.3.1.7, 5.3.2.9, 5.3.3, 5.3.13, 5.3.15, 5.4.4.1 Fig. 1 § 5.2.9, 5.3.16, 5.3.9, 5.3.1, 5.3.3, 5.3.15, 5.4.3, 5.5.1, § 5.3.1, 5.3.3, 5.3.10, 5.3.17, 5.4.3, 5.5, 5.2.7, 5.3.17, 5.6.1.8 § 7 § 3.2, 3.4, 4.3.1, 4.4.1. § 3.3, 3.4	Update according EUG comments on V. 0.2.0 Update key generation request Suppression of abort message Interface to coordination layer ENISA recommendation Update terms and abbreviation lists And corrections marked ‘Editorial’	WG
0.3.1 18-11-2015	3.2, 3.3, 3.4, 7.1.1.1, 7.2	Update as per 4 November 2015 EECT meeting.	PP
0.3.2 09-12-2015	§ 3.2, 3.4.1.1, 4.2.6.3, 5.3.1.4, 5.3.2.7, 5.3.3, 5.3.4.2, 5.3.9, 6.3.2.5, 7.1.1.2 and 7.4	Update as per consolidated review sheet of the 08-12-2015 EECT meeting	FH

	Fig. 5 and 10		
1.0.0 17-12-2015	-	Baseline 3 2nd release version	PP

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 and, if needed, for the generation and checking of certificates to guarantee the authenticity of the communicating entities.

3.2 References

[ENISA]	Algorithms, key size and parameters report 2014	November 2014
[ENISA_1]	Study on cryptographic protocols	November 2014
[EN-50159]	Safety-related communication in transmission systems	September 2010
[RFC-1320]	The MD4 Message-Digest Algorithm	April 1992
[RFC-2560]	X.509 Internet Public Key Infrastructure Online Certificate Status Protocol – OCSP	June 1999
[RFC-4055]	Additional Algorithms and Identifiers for RSA Cryptography for use in the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile	June 2005

[RFC-4210]	Internet X.509 Public Key Infrastructure Certificate Management Protocol (CMP)	September 2005
[RFC-4211]	Internet X.509 Public Key Infrastructure Certificate Request Message Format (CRMF)	September 2005
[RFC-4279]	Pre-Shared Key Ciphersuites for Transport Layer Security (TLS)	December 2005
[RFC-5246]	The Transport Layer Security (TLS) Protocol. Version 1.2	August 2008
[RFC-5280]	Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile	May 2008
[RFC-5487]	Pre-Shared Key Cipher Suites for TLS with SHA-256/384 and AES Galois Counter Mode	March 2009
[RFC-6277]	Online Certificate Status Protocol Algorithm Agility	June 2011
[RFC-6818]	Updates to the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile	January 2013
[RFC-6961]	The Transport Layer Security (TLS) Multiple Certificate Status Request Extension	June 2013
[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	
[X.500]	ITU-T Recommendation: Information technology - Open Systems Interconnection - The Directory: Overview of concepts, models and services	October 2012
[X.501]	ITU-T Recommendation: Information technology - Open Systems Interconnection - The Directory: Models	October 2012
[X.520]	ITU-T Recommendation: Information technology - Open Systems Interconnection - The Directory: Selected attribute types	October 2012
[X.690]	ITU-T Recommendation: Information technology - ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)	July 2002

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
CA	Certificate Authority
CMP	Certificate Management Protocol
DB	DataBase
DN	Distinguished Name
ECC	Elliptic Curve Cryptography
ECDH	Elliptic Curve Diffie Hellman
OCSP	On-line Certificate Status Protocol
PSK	Pre-Shared Key
RA	Registration Authority
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
Certificate Authority (CA)	The entity responsible for issuing digital certificates to associate public keys with user identities
Confidentiality	Confidentiality, in the context of computer systems, allows only authorised users to access protected data using specific mechanisms to ensure confidentiality and safeguard data from harmful intrusion
Cryptography	The principles, means and methods for transformation of data in order to ensure confidentiality, authenticity, non-repudiation and integrity
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.
Registration Authority (RA)	The responsible entity in a PKI for accepting requests for digital certificates and authenticating the entity making the request

Security Infrastructure	The set of hardware, software, people, policies and procedures needed to manage the registration of entities and distribution and storage of digital certificates in a system
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.

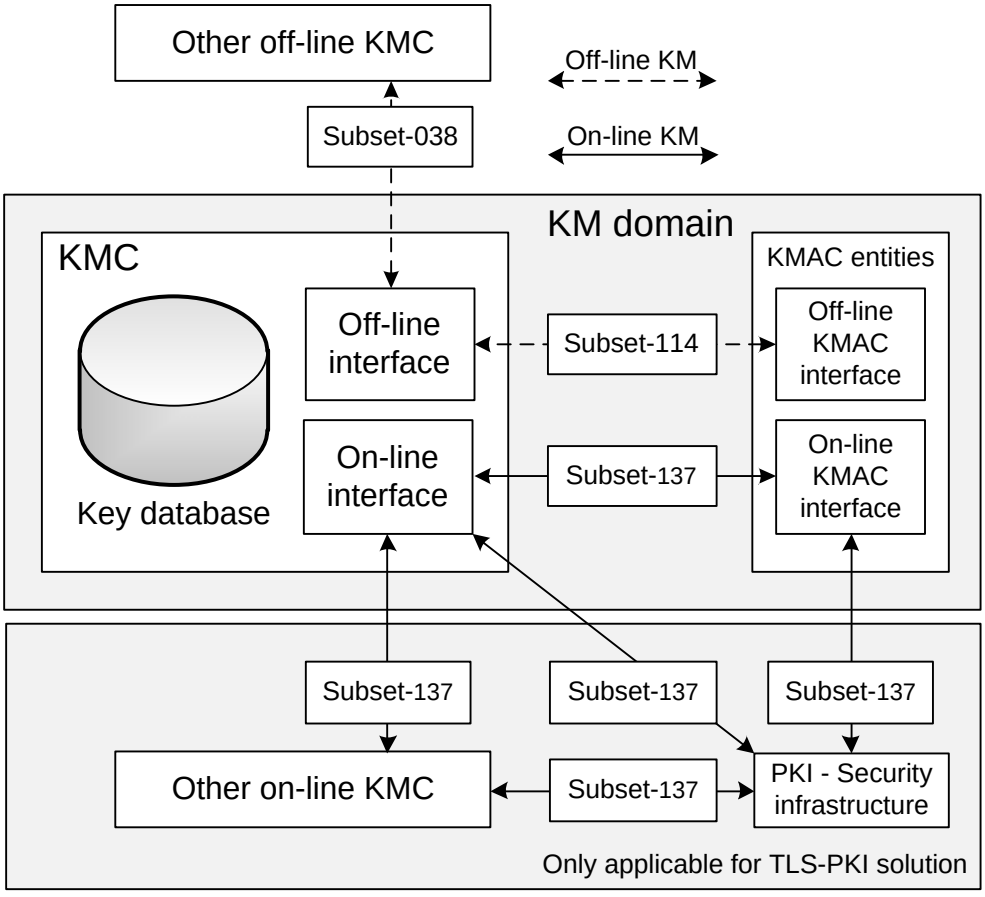

Figure 1 – KMS Reference Architecture

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.

4.3.1.2 The authentication shall be guaranteed either by using certificates from a Public Key Infrastructure (PKI) or by using secret pre-shared keys (PSK).

4.3.1.3 The TLS protocol using pre-shared keys for authentication is referred to as TLS-PSK throughout the rest of the document.

4.3.1.4 The TLS protocol using a Public Key Infrastructure for authentication is referred to as TLS-PKI throughout the rest of the document.

4.3.1.4.1 Note: Even if this standard specifies the use of one of the strongest cipher suites for TLS, this cipher suite is only recommended for legacy use due to the lack of security proof (see table 2.1 of [ENISA]) in the ENISA study on cryptographic protocol (see [ENISA_1]). Using TLS and PKI, there is no alternative cipher suite available yet.

4.3.1.5 Every KMS entity shall support TLS-PKI.

4.3.1.6 A KMS entity may optionally support TLS-PSK as an alternative to TLS-PKI, but only for use within a KM domain and not between two KMC.

4.3.1.6.1 Note: TLS-PSK may be more convenient from a key management point of view in small KM domains. In such domains it may be easier to use pre-shared keys than to set up a public key infrastructure. TLS-PSK may also be used as fall-back in case TLS-PKI is not available, e.g. due to a compromised CA.

4.3.1.7 TLS-PSK relies on the secrecy of a pre-shared key to authenticate the peer entity. The distribution and installation of secret pre-shared keys must be supported by operational procedures to guarantee the secrecy and authenticity. The definition of these operational procedures is out of scope of this document.

4.3.1.8 TLS-PKI relies on digital certificates managed and distributed by an external Certificate Authority (CA) to authenticate the peer entity. The certificate of the root CA must be installed in all peer entities using some operational procedures to guarantee its authenticity. The definition of these operational procedures is out of scope of this document.

4.3.1.9 The following figure depicts the general PKI certificate hierarchy.

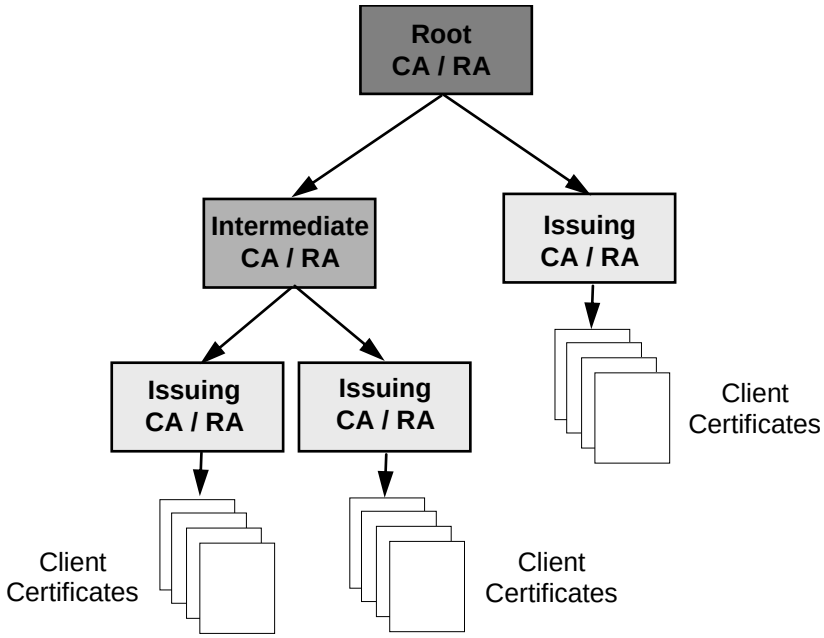

Figure 2 – PKI certificate hierarchy

4.3.1.10 In the figure above, CA is a Certificate Authority responsible for issuing, renewing and revoking digital certificates. A digital certificate contains, among others, a public key and information related to the key, its owner, its validity period and its allowed use (e.g. encryption and/or authentication).

4.3.1.11 In the simplest scenario, certificates are issued by a Certificate Authority. More complex scenarios see the presence of a Registration Authority (RA). When a new

entity wants to obtain a client certificate, it issues a request to the RA which then tries to authenticate the requester. If authenticated, the RA forwards the request to the CA, which issues the digital certificate. The RA can be a part of a CA as well as a separate entity.

4.3.1.12 CAs can be organized in a hierarchical tree structure with CA certificates issued by a higher-level CA. This tree structure has a single root node, called “root CA” and all clients must know the root CA certificate.

4.3.1.13 Digital certificates are distributed using CMP.

4.3.1.14 The digital certificate is validated by using OCSP or by using the “Multiple Certificate Status Request” extension of the TLS protocol.

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:

a. generating the public/private key pair (see § 6.3.1.4.3);
b. generating the pre-shared key used if the TLS-PSK solution is used (see § 6.2.3.1);
c. generating the KMAC (see § 5.2.2.1);
d. performing the TLS handshake procedure (see § 6.2.4).

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	→	KMS entity
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	→	KMC
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 3 – 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..100]	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.

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.
		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”.
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

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, this connection shall be released by the KMS entity detecting the time-out.

5.4.4.2 If the application time-out elapses for a connection established at application level, this connection shall be released by the KMS entity detecting the time-out.

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 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.

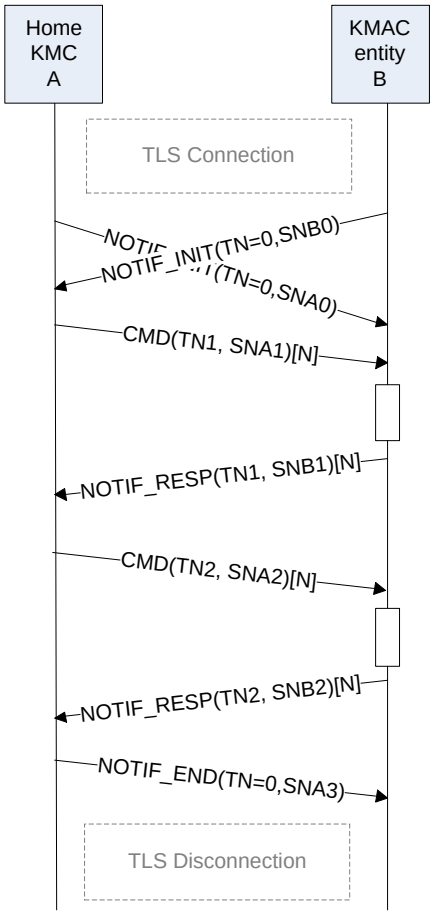

Figure 4 – KMC-KMAC entity key management scenario

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.

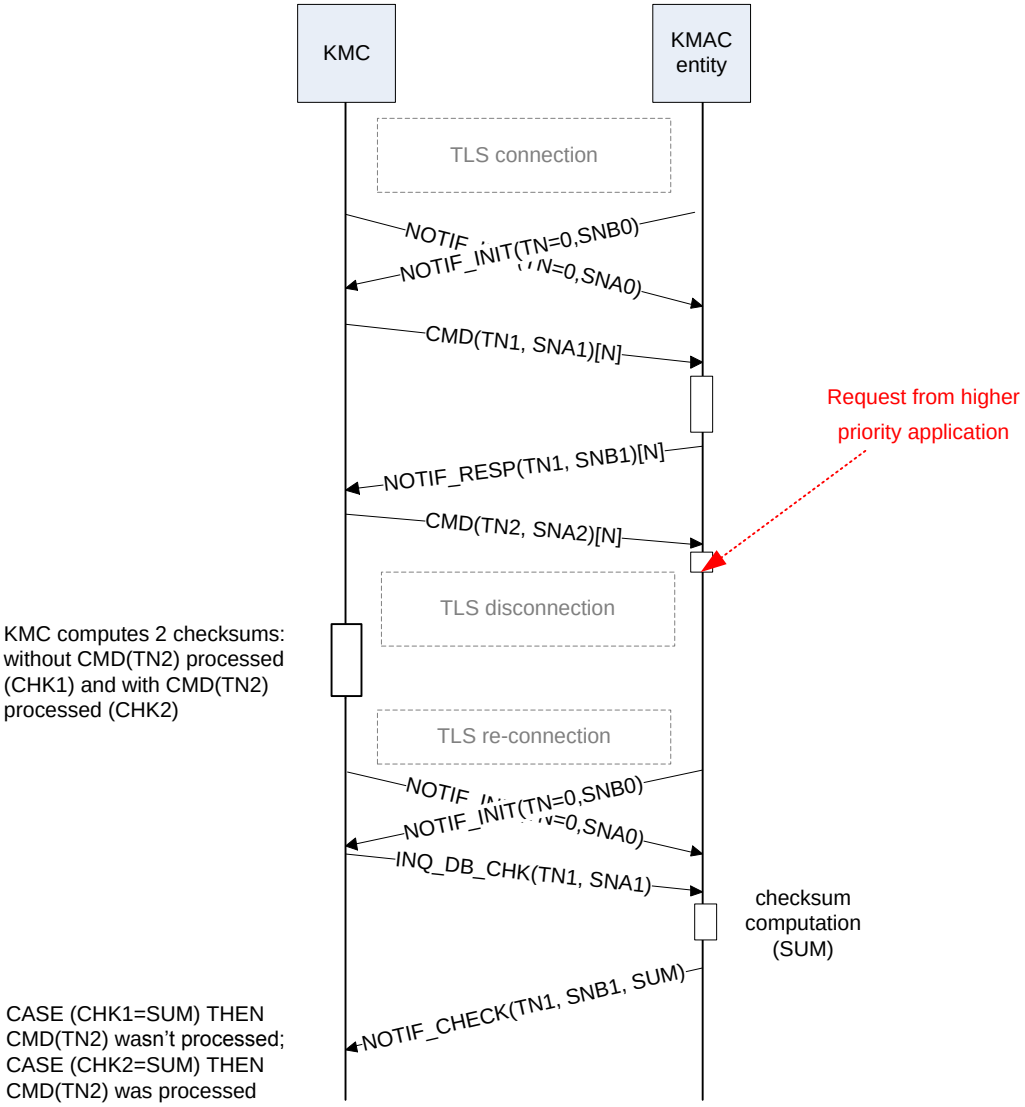

Figure 5 – KMC-KMAC entity: abnormal session release

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 sends NOTIF_SESSION_ABORT and 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.

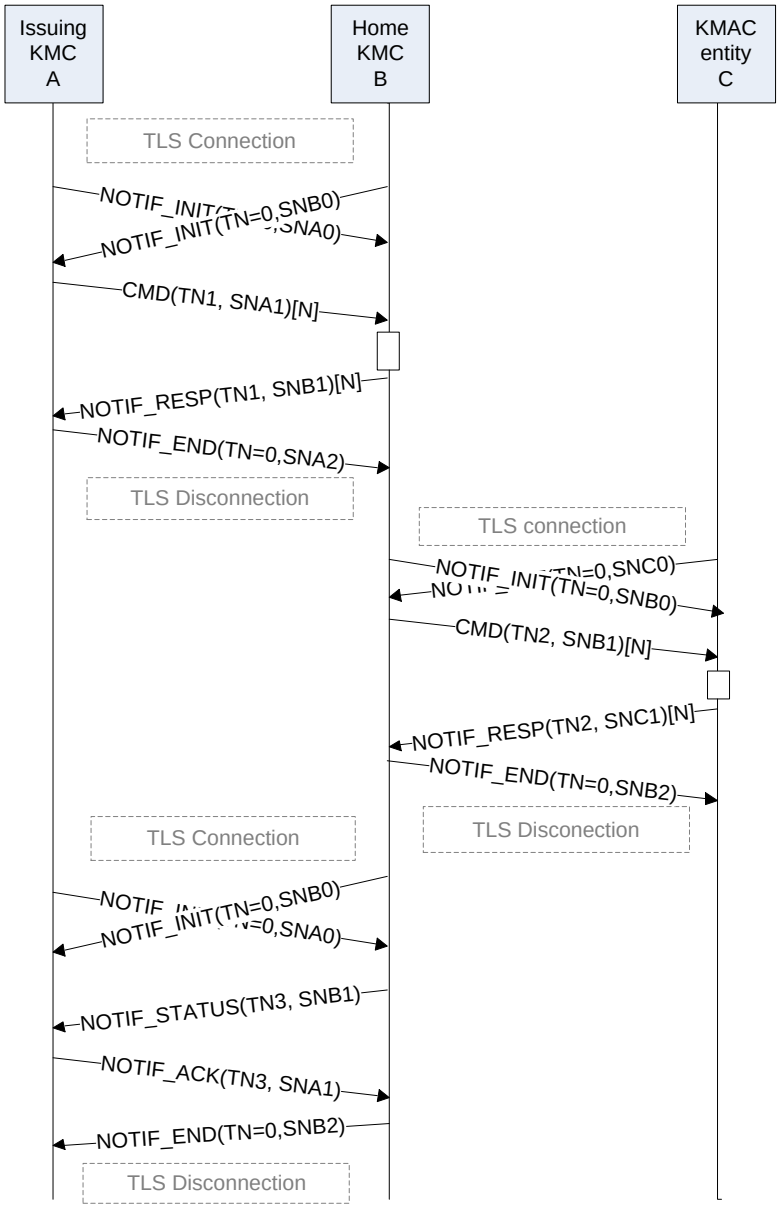

Figure 6 – KMC-KMC key management scenario

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.

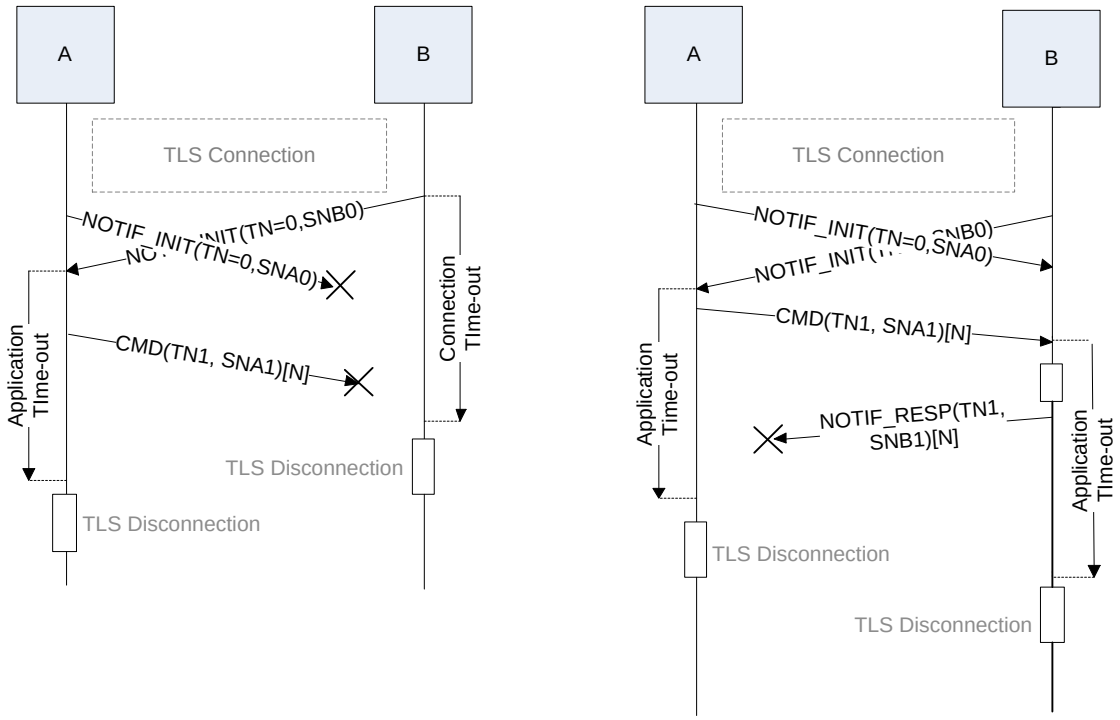

Figure 7 – Time-out supervision scenarios

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.

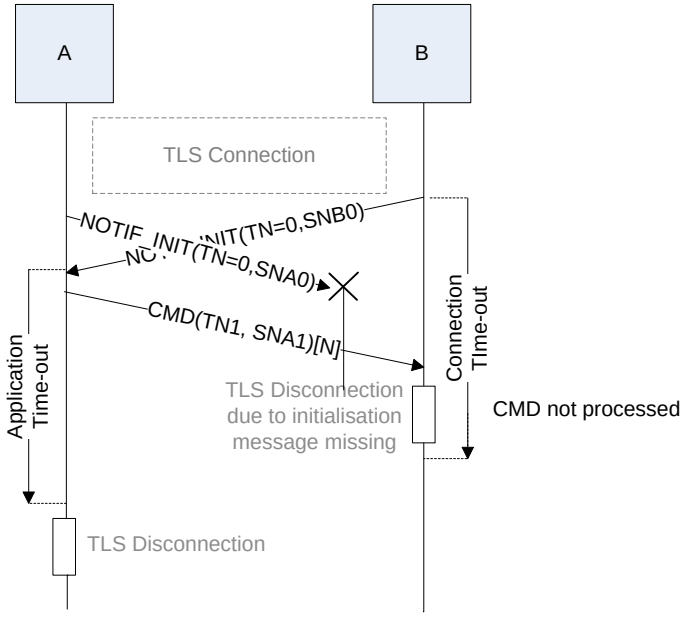

Figure 8 – Sequence error during connection establishment

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.

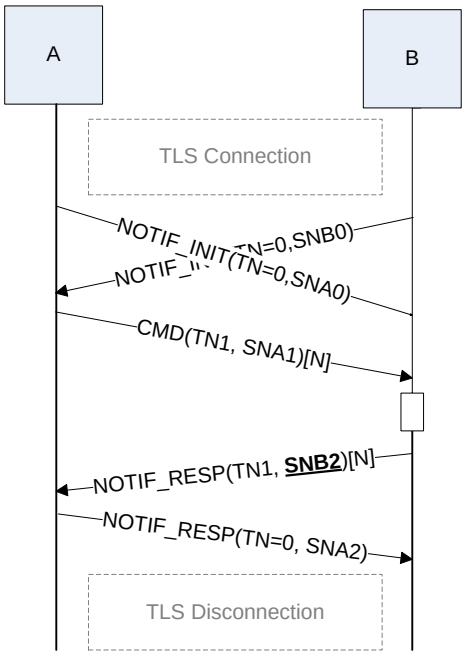

Figure 9 – Sequence number error scenario

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.

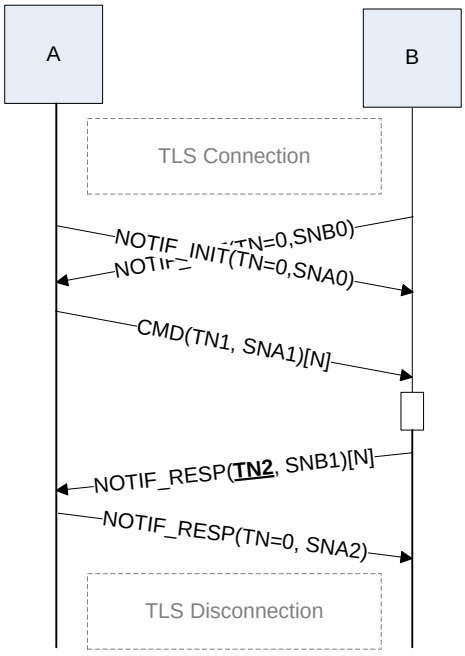

Figure 10 – Transaction number error scenario

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:

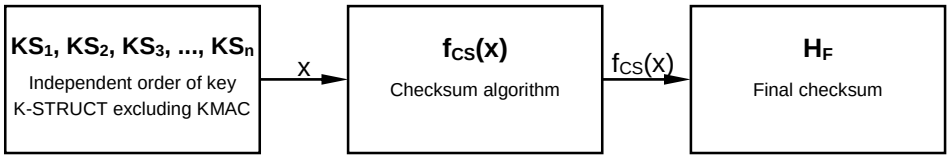

Figure 11 – Overview of the Key DB checksum algorithm

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:

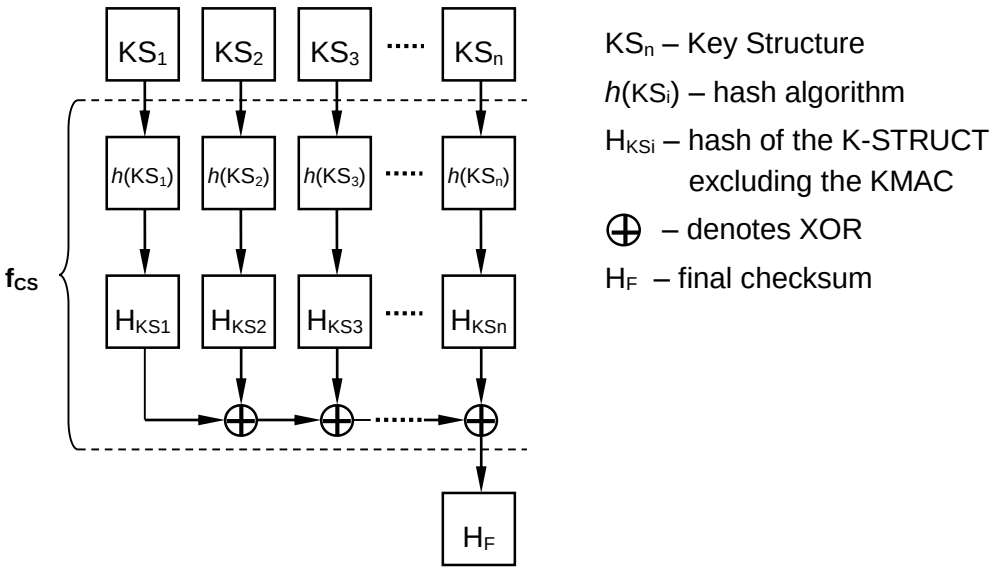

Figure 12 – Definition of the checksum algorithm

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 K-STRUCT excluding the KMAC as 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

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 Scope and purpose

6.1.1.1 This chapter specifies the following interfaces:

a) TLS interface: this interface allows establishing a TLS connection between two KMS entities and securely exchange information over this connection.

b) Certificate delivery interface: this interface between a PKI client (KMS entity) and a Certificate Authority allows generating or renewing the certificate of the PKI client using the CMP protocol.

c) Certificate status management: this interface between an OCSP client (KMS entity) and an OCSP responder (Certificate Authority) allows to check the validity of a peer entity certificate using the OCSP protocol.

6.1.1.2 This chapter lists the necessary information required to define the TLS, CMP and OCSP protocols in order to:

a) establish a TLS connection between two KMS entities;

b) exchange information protected by the established TLS connection;

c) request for a new certificate;

d) check the validity of certificates from KMS entities.

6.2 TLS interface specification

6.2.1 Role allocation

6.2.1.1 The TLS client is the entity responsible for establishing the connection with the TLS server.

6.2.1.2 The KMC shall implement the TLS server function for the following connections:

a) KMC-KMC

b) KMC-KMAC on-board entity

6.2.1.3 The KMC shall implement the TLS client function for the following connections:

a) KMC-KMC

b) KMC-KMAC trackside entity

6.2.1.4 The KMAC trackside entity shall implement a TLS server.

6.2.1.5 The KMAC on-board entity shall implement a TLS client.

6.2.2 TLS common requirements

6.2.2.1 The TLS protocol is used in two phases:

a) Handshake phase for authenticating the client and server, and negotiating the cryptographic information (Premaster secret, algorithm, etc.) necessary for establishing the TLS session

b) Application data phase (TLS session) for securely exchanging data using the keys and algorithms negotiated during the handshake phase

6.2.2.2 The TLS version 1.2 shall be supported (see [RFC-5246]). Older versions (1.1, 1.0) shall not be supported.

6.2.2.3 TLS communication between KMS entities shall be authenticated.

6.2.2.4 TLS communication between KMS entities shall be encrypted.

6.2.2.5 TLS communication between KMS entities shall not use a compression algorithm.

6.2.2.6 Resumption of a previous TLS session, duplication of an existing TLS session and renegotiation of an existing TLS session is not allowed.

6.2.3 TLS requirements for TLS-PSK

6.2.3.1 A unique pre-shared key shall be generated by the KMC for each pair of KMC and KMAC entity. This pre-shared key shall be used to authenticate both peers.

6.2.3.2 Installation of a pre-shared key in a KMS entity overwrites any previously stored pre-shared key in this KMS entity.

6.2.3.3 The size of the pre-shared key shall be at least 256 bits.

6.2.3.4 The TLS clients and servers shall support at least the following cipher suite: TLS_DHE_PSK_WITH_AES_256_GCM_SHA384 (see [RFC-4279], [RFC-5487]).

6.2.3.5 If other pre-shared key cipher suites are supported by TLS clients and servers, these cipher suites must be recommended by [ENISA] for ‘Future System Use’.

6.2.3.6 The minimum parameter sizes of other supported cipher suite algorithms shall be compliant with the recommendation made by ENISA for ‘Future System Use’ (see [ENISA] § 3.6).

6.2.3.7 The handshake procedure, as well as the use of the TLS protocol messages for the cipher suites defined above, is specified in detail in [RFC-4279] and [RFC-5487].

6.2.3.8 As both TLS clients and TLS servers may have pre-shared keys with different peers, it is necessary to know which key to use. Therefore, the TLS client indicates which key to use by including a “PSK identity” in the ClientKeyExchange message (see [RFC-4279], §2). In addition to help the client in selecting which identity to use, the server shall provide a PSK identity hint in the ServerKeyExchange message (see [RFC-4279], §2).

6.2.3.9 The expanded ETCS ID of the sender of the TLS message shall be used as the PSK identity and PSK identity hint.

6.2.4 TLS requirements for TLS-PKI

6.2.4.1 The authentication shall be mutual between the TLS client and server, and based on X509 v.3 certificates (see [RFC-5280]) delivered through the KMS's PKI.

6.2.4.2 The TLS clients and servers shall support the following cipher suite: TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (see [RFC-5289]).

6.2.4.3 If other cipher suites are supported by the TLS clients and servers, these cipher suites must be recommended by [ENISA] for 'Future System Use'.

6.2.4.4 The minimum parameter sizes of other supported cipher suite algorithms shall be compliant with the recommendation made by ENISA for 'Future System Use' (see [ENISA] § 3.6).

6.2.4.5 The TLS protocol defines some optional messages and for each message the potential extensions. The table below specifies for the selected cipher suite the list of supported messages, the direction of the message and for each message which extensions or options are supported, if any. Other messages are not supported and shall not be used.

Message	Message flow direction			Extensions or options used
Client Hello	TLS client	→	TLS server	Both "Elliptic Curves Extension" and "Supported Point Formats Extension" should be used (see [RFC-4492]). "Multiple Certificate Status Request" extension should be used (see [RFC-6961]).
Server Hello	TLS server	→	TLS client	Both the "Elliptic Curves Extension" and the "Supported Point Formats Extension" shall be supported. "Multiple Certificate Status Request" extension should be used (see [RFC-6961]).
Server Certificate	TLS server	→	TLS client	No extension or option could be used.
Certificate Status	TLS server	→	TLS client	Optional message sent to provide the list of OCSP responses for certificates as answer to the "Multiple Certificate Status Request" extension.
Server Key Exchange	TLS server	→	TLS client	No extension or option could be used.
Certificate Request	TLS server	→	TLS client	Extended as specified in § 5.5 of [RFC-4492].
Server Hello Done	TLS server	→	TLS client	No extension or option could be used.
Client Certificate	TLS client	→	TLS server	No extension or option could be used.

Message	Message flow direction			Extensions or options used
Client Key Exchange	TLS client	→	TLS server	Extended as specified in the § 5.7 of [RFC-4492]
Certificate Verify	TLS client	→	TLS server	No extension or option could be used.
Finished	TLS client	↔	TLS server	No extension or option could be used.

Table 2: TLS messages for selected cipher suite

6.2.4.6 The optional use of “Multiple Certificate Status Request” extension (see [RFC-6961]) is foreseen in order to:

a) mitigate the risk of CA on-line unavailability, mainly for wireless connections;

b) speed up the TLS connection establishment time, mainly for wireless connections;

c) reduce the number of connections to the CA.

6.2.4.7 **Hello messages**

6.2.4.7.1 The TLS client and TLS server shall support and use the optional “Server Hello” and “Client Hello” messages.

6.2.4.7.2 A TLS client that proposes ECC cipher suites may choose not to include the “Elliptic Curves Extension” and “Supported Point Formats Extension”. In this case, the server is free to choose the elliptic curves or point formats.

6.2.4.7.3 The TLS client and TLS server shall at least support the elliptic curve “brainpoolP256r1” (see [RFC-5639]).

6.2.4.7.4 A TLS client that proposes “Multiple Certificate Status Request” extension may provide a zero-length "responder_id_list". In this case, the responders must be implicitly known by the server, or must be identified by the certificates used by the server.

6.2.4.7.5 The TLS server shall support “Elliptic Curves Extension” and “Supported Point Formats Extension”.

6.2.4.7.6 The “Supported Point Formats Extension” shall be included in a Server Hello message in response to a Client Hello message containing the “Supported Point Formats Extension” when negotiating an ECC cipher suite.

6.2.4.7.7 A TLS server should support “Multiple Certificate Status Request” extension. In such case the server shall return an extension of type "status_request_v2" with empty "extension_data".

6.2.4.8 **Server Certificate**

6.2.4.8.1 The optional “Server Certificate” message shall be used by the TLS server and supported by the TLS client to authenticate the server.

6.2.4.9 **Certificate Status**

6.2.4.9.1 The “Certificate Status” message shall be used by the TLS server in order to report the list of OCSP responses for the matching corresponding certificate in the server Certificate in case of use of “Multiple Certificate Status Request” extension.

6.2.4.9.2 The periodicity for refreshing the list of OCSP responses is a TLS server configuration parameter. This time period value shall be between 1 hour and 100 hours, with the default value being 10 hours.

6.2.4.9.3 In case of successful use of the “Multiple Certificate Status Request” extension, including a freshness check, the TLS client does not need to check the certificate status of the peer entities through OCSP requests.

6.2.4.10 **Server Key Exchange**

6.2.4.10.1 The “Server Key Exchange” message shall be used by the TLS server and supported by the TLS client to convey the server’s ephemeral ECDH public key (and the corresponding elliptic curve domain parameters) to the client.

6.2.4.11 **Certificate Request**

6.2.4.11.1 The “Certificate Request” message shall be used by the TLS server and supported by the TLS client.

6.2.4.11.2 This message shall be extended as specified in § 5.5 of [RFC-4492].

6.2.4.12 **Server Hello Done**

6.2.4.12.1 The “Server Hello Done” message shall be used by the TLS server and supported by the TLS client.

6.2.4.13 **Client Certificate**

6.2.4.13.1 The optional “Client Certificate” message shall be used by the TLS client and supported by the TLS server.

6.2.4.13.2 The “Client Certificate” message shall comply with the certificate types listed in the Certificate Request.

6.2.4.14 **Client Key Exchange**

6.2.4.14.1 The optional “Client Key Exchange” message shall be used by the TLS client and supported by the TLS server.

6.2.4.14.2 This message shall be extended as specified in § 5.7 of [RFC-4492].

6.2.4.15 **Certificate Verify**

6.2.4.15.1 The optional “Certificate Verify” message shall be used by the TLS client and supported by the TLS server.

6.2.4.16 **Finished**

6.2.4.16.1 The “Finished” message shall be supported and used by both the TLS client and the TLS server.

6.3 Certificate delivery interface

6.3.1 Client certificate delivery functions

6.3.1.1 The following table provides the function allocation for the certificate delivery interface:

Function	Message flow direction			Purpose
Certificate Request	PKI client	→	PKI server	Request a certificate
Certificate Response	PKI server	→	PKI client	Deliver a certificate
Certificate Confirmation	PKI client	→	PKI server	Confirm the reception of a certificate
Confirmation Acknowledgement	PKI server	→	PKI client	Acknowledge the Certificate Confirmation message

Table 3: Functions allocation for the certificate delivery interface

6.3.1.2 Three kinds of certificate generation exist:

a) First certificate: a new certificate with a new public key (first request without any valid certificate);

b) Certificate renewal: a new certificate with the same key;

c) Certificate rekey: a new certificate with a new key (when requester already has a valid certificate).

6.3.1.3 For KMS purposes, only requests for a first certificate or certificate rekey shall be used.

6.3.1.4 **Certificate Request**

6.3.1.4.1 The PKI client is responsible to request the PKI server for delivery of a first certificate.

6.3.1.4.2 The PKI client is responsible to request a certificate rekey a configurable time before the expiration of the current certificate.

6.3.1.4.3 The PKI client shall generate a public/private key pair at first certificate request and at every certificate rekey request.

6.3.1.4.4 The PKI client is responsible for keeping its private key secret.

6.3.1.4.5 The public key length for a PKI client shall be 3072 bits.

6.3.1.4.6 The public key length for a Certificate Authority shall be 3072 bits.

6.3.1.4.7 Each PKI client shall have its own Distinguished Name (DN). This DN is unique in the KMS (see section 6.3.3).

6.3.1.4.8 In case of a first certificate request, the PKI client shall authenticate itself by using shared secret information (a ‘passphrase’) to create the “protection” field contained in the “Certificate Request” message.

6.3.1.4.9 The characters used for the passphrase shall be encoded using UTF-8 with a minimum length of 16 characters.

6.3.1.4.10 The ‘passphrase’ shall not be part of the initial configuration but shall be provided when needed by a specific process independent from this interface.

6.3.1.4.11 In case of a certificate rekey request, the PKI client has a valid certificate and shall use this valid certificate to authenticate itself when requesting a new certificate. The PKI client shall create the “protection” field contained in the “Certificate Request” message by using the private key associated to its valid certificate.

6.3.1.5 Certificate Response

6.3.1.5.1 If the PKI server considers a certificate request from a PKI client as valid, the PKI server shall be able to sign and deliver a new certificate to this PKI client.

6.3.1.5.2 If a certificate request is valid, the “Certificate Response” message shall include:

a) a signed certificate corresponding to the template certificate contained in the certificate request;

b) the certificate hierarchy for this certificate, except the root certificate which has to be delivered to the PKI client in a secure way (with an organisational process).

6.3.1.5.3 If the certificate request is not valid, the PKI server shall send a negative certificate response.

6.3.1.6 Certificate Confirmation

6.3.1.6.1 A “Certification Confirmation” message shall be sent by the PKI client to the PKI server at reception of the “Certificate Response” message.

6.3.1.7 Confirmation Acknowledgment

6.3.1.7.1 The PKI server shall acknowledge the reception of the confirmation from the PKI client.

6.3.1.7.2 Both the certificate in use for a certificate rekey and the new certificate are valid up to the end of their own validity period, unless revoked by the Certificate Authority.

6.3.2 Interface specification

6.3.2.1 General requirements

6.3.2.1.1 The “Certificate Request”, “Certificate Response”, “Certification Confirmation” and “Confirmation Acknowledgement” messages shall be exchanged in the same TCP/IP session.

6.3.2.1.2 In case of TCP disconnection during the certificate distribution process, both PKI client and PKI server shall discard the current operation, the process shall be considered as failed by both and the PKI server shall revoke the newly generated certificate.

6.3.2.1.3 The certificate delivered by the PKI server shall conform to X509 v.3 (see [RFC-5280] and [RFC-6818]).

6.3.2.1.4 The “Certificate Request”, “Certificate Response”, “Certification Confirmation” and “Confirmation Acknowledgement” messages shall comply with the CMP protocol (see [RFC-4210] and [RFC-4211]).

6.3.2.1.5 For each optional field in the CMP messages, it shall be stated if this optional field:

- shall be used
- shall not be used
- can be used

6.3.2.1.6 The used fields shall be present in the message.

6.3.2.1.7 The fields that are not used shall not be present in the message.

6.3.2.2 CMP message specification: PKI message common fields

6.3.2.2.1 The “Certificate Request”, “Certificate Response”, “Certification Confirmation” and “Confirmation Acknowledgement” messages shall comply with Table 4 and Table 5.

PKIMessage (see RFC-4210)		
Header	Mandatory	See PKIHeader defined in Table 5
Body	Mandatory	Shall be one of the following: 1. CertReqMessages (defined in Table 6) for a Certificate Request Message 2. CertRepMessage (defined in Table 12) for a Certificate Response Message 3. CertConfirmContent (defined in Table 18) for a Certificate Confirmation Message 4. PKIConfirmContent (see section 6.3.2.6) for a Confirmation Acknowledgement Message
Protection	Optional	Shall be used
extraCerts 1..MAX	Optional	Shall be used

Table 4: PKIMessage

PKIHeader (See RFC-4210)		
pvno	Mandatory	
sender	Mandatory	Distinguished Name shall be used
recipient	Mandatory	Distinguished Name shall be used
messageTime	Optional	Can be used UTC time shall be chosen
protectionAlg	Optional	Shall be used For a first certificate request, algorithmIdentifier shall be PasswordBasedMAC for messages emitted by the PKI client and sha384WithRSAEncryption for messages emitted by the PKI server ([RFC-4055]).

		For a certificate rekey, the algorithmIdentifier shall be sha384WithRSAEncryption for PKI client and server.
senderKID	Optional	Shall be used for the messages emitted by the PKI client for rekey purpose. Can be used for other purposes
recipKID	Optional	Can be used
transactionID	Optional	Shall be used
senderNonce	Optional	Shall be used
recipNonce	Optional	Shall be used
freeText	Optional	Shall not be used
generalInfo 1..MAX	Optional	Shall not be used

Table 5: PKIHeader

6.3.2.3 CMP message specification: “Certificate Request” message

6.3.2.3.1 For the “Certificate Request” message, the body of the PKI message shall comply with Table 6, Table 7, Table 8, Table 9, Table 10 and Table 11.

CertReqMessage (see RFC-4211)		
certReqMsg 1..MAX	Mandatory	See CertReqMsg defined in Table 7

Table 6: CertReqMessage

CertReqMsg (see RFC-4211)		
certReq	Mandatory	See CertRequest defined in Table 8
popo	Optional	Shall be used. Shall be a signature of type POPOSigningKey (see Table 11)
regInfo 1..MAX	Optional	Shall not be used

Table 7: CertReqMsg

CertRequest (see RFC-4211)		
certReqId	Mandatory	
certTemplate	Mandatory	See CertTemplate defined in Table 9
controls	Optional	Shall not be used

Table 8: CertRequest

CertTemplate (see RFC-4211)		
version	Optional	Shall be used
serialNumber	Optional	Shall not be used
signingAlg	Optional	Shall not be used
issuer	Optional	Shall not be used
validity	Optional	Shall not be used
subject	Optional	Shall be used
publicKey	Optional	Shall be used See SubjectPublicKeyInfo defined in Table 10
issuerUID	Optional	Shall not be used
subjectUID	Optional	Shall not be used
extensions	Optional	Shall not be used

Table 9: CertTemplate

SubjectPublicKeyInfo (see RFC-3280 and RFC-5280)		
algorithm	Mandatory	The AlgorithmIdentifier shall be rsaEncryption
subjectPublicKey	Mandatory	

Table 10: SubjectPublicKeyInfo

POPOSigningKey (see RFC-4211)		
poposkInput	Optional	Shall not be used
algorithmIdentifier	Mandatory	The algorithmIdentifier shall be sha384WithRSAEncryption ([RFC-4055])
signature	Mandatory	

Table 11: POPOSigningKey

6.3.2.4 CMP message specification: “Certificate Response” message

6.3.2.4.1 For the “Certificate Response” message, the body of the PKI message shall comply with Table 12, Table 13, Table 14, Table 15, Table 16 and Table 17.

CertRepMessage (see RFC-4210)		
caPubs 1..MAX	Optional	Shall be used
response	Mandatory	See CertResponse in Table 13

Table 12: CertRepMessage

CertResponse (see RFC-4210)		
certReqId	Mandatory	
status	Mandatory	
certifiedKeyPair	Optional	Shall be used See CertifiedKeyPair defined in Table 14
rspInfo	Optional	Shall not be used

Table 13: CertResponse

CertifiedKeyPair (see RFC-4210)		
certOrEncCert	Mandatory	See CertOrEncCert in Table 15
privateKey	Optional	Shall not be used
publicationInfo	Optional	Shall not be used

Table 14: CertifiedKeyPair

CertOrEncCert (see RFC-4210)		
certificate or encryptedCert	Mandatory	Certificate shall be chosen

Table 15: CertOrEncCert

Certificate (see RFC-5280)		
tbsCertificate	Mandatory	See TBSCertificate defined in Table 17
signatureAlgorithm	Mandatory	The algorithmIdentifier shall be sha384WithRSAEncryption
signatureValue	Mandatory	

Table 16: Certificate

TBSCertificate (see RFC-5280)		
version	Mandatory	
serialNumber	Mandatory	
signature	Mandatory	The algorithmIdentifier shall be sha384WithRSAEncryption
issuer	Mandatory	
validity	Mandatory	UTC time shall be chosen
subject	Mandatory	

subjectPublicKeyInfo		Mandatory	See SubjectPublicKeyInfo defined in Table 10
issuerUniqueID		Optional	Shall not be used
subjectUniqueID		Optional	Shall not be used
Certificate extensions	Authority Key Identifier	Mandatory for not « self-signed » conforming CA.	Shall be used
	Subject Key Identifier	Mandatory for conforming CA	Shall be used
	Key Usage	Optional	Shall be used. Key usage shall indicate at least that certificate MUST allow the key to be used for signing
	Certificate Policies	Optional	Shall not be used
	Policy Mappings	Optional	Shall not be used
	Subject Alternative Name	Optional	Shall not be used
	Issuer Alternative Name	Optional	Shall not be used
	Subject Directory Attributes	Optional	Shall not be used
	Basic Constraints	Conforming CAs MUST include this extension in all CA certificates that contain public keys used to validate digital signatures on certificates and MUST mark the extension as critical in such certificates.	Shall be used
	Name Constraints	Optional	Shall not be used
	Policy Constraints	Optional	Shall not be used
	Extended Key Usage	Optional	Shall not be used
	CRL Distribution Points	Optional	Shall not be used
	Inhibit anyPolicy	Optional	Shall not be used
	Freshest CRL	Optional	Shall not be used
	Authority Information Access	Optional	Shall be used The authority information access extension indicates how to access information and services for the issuer of the certificate in which the extension appears. The LDAP protocol shall be used to access the information.
	Subject Information Access	Optional	Shall not be used

Table 17: TBSCertificate

6.3.2.5 CMP message specification: “Certification Confirmation” message

6.3.2.5.1 For the “Certificate Confirmation” message, the body of the PKI message shall comply with Table 18.

CertConfirmContent (see RFC-4210)		
certHash	Mandatory	
certReqId	Mandatory	
statusInfo	Optional	Shall not be used

Table 18: CertConfirmContent

6.3.2.6 CMP message specification: “Confirmation Acknowledgement” message

6.3.2.6.1 For the “Confirmation Acknowledgement” message, the body of the PKI message shall be empty (see PKIConfirmContent in RFC-4210).

6.3.3 Distinguished Name

6.3.3.1 A Distinguished Name is a name given to an element within a computer system or a network that uniquely identifies it.

6.3.3.2 The Distinguished Name syntax is defined in standards [X.520], [X.500] and [X.501].

6.3.3.3 A Distinguished Name is made up of “attribute=value” pairs, separated by commas.

6.3.3.4 The on-line KMS will use names with the attributes in the order stated here below:

Distinguished Name		
Key identifier	Attribute type	Content
C	Country Code	ISO alpha-2 country code
O	Organization Name	Acronym of the organisation operating the element identified by the OID. This acronym shall be composed of 2 or 3 uppercase characters from the Latin alphabet [ISO-8859-1].
OU	Organizational Unit Name	Element abbreviation has to be used as Unit name. I.e. one of the following: • KMC • RBC • EVC • RIU • RA • CA
CN	Common Name	The common name represents the name given to the element. This name shall be composed at maximum of 32 upper case characters from the Latin alphabet [ISO-8859-1] or digits. For the KMC, RBC, EVC, and RIU the Common Name shall be the expanded ETCS ID in hexadecimal.

Table 19: Distinguished Name syntax

6.3.3.5 For example, an ETCS on-board equipment (EVC) with ETCS-ID-EXP: 02E6A54B (where 02 is the ETCS-ID type and E6A54B is the ETCS-ID) operated by Banedanmark (BDK) in Denmark (DK) shall have the following Distinguished Name:

DK, BDK, EVC, 02E6A54B

6.4 Certificate status check interface

6.4.1 Certificate status check functions

6.4.1.1 The following table provides the functions allocation for the certificate status check interface:

Function	Allocation	Purpose
OCSP Request	PKI client	Request for the revocation status of a certificate
OCSP Response	PKI server	Provide the revocation status of a certificate

6.4.1.2 **OCSP Request**

6.4.1.2.1 The PKI client shall be able to check whether the certificate of a peer has been revoked or not, by sending an OCSP Request to the PKI server.

6.4.1.3 **OCSP Response**

6.4.1.3.1 The PKI server shall be able to send an OCSP Response to a PKI client having emitted an OCSP Request.

6.4.2 Interface specification

6.4.2.1 **General requirements**

6.4.2.1.1 The OCSP Request and OCSP Response shall be exchanged on the same TCP/IP session.

6.4.2.1.2 In case of TCP disconnection during the certificate check process, both PKI client and PKI server shall discard the current operation and consider the process as failed.

6.4.2.1.3 The OCSP Request and OCSP Response messages shall conform to OCSP protocol described in [RFC-2560] and [RFC-6277].

6.4.2.1.4 The check of the peer certificate chain shall be performed at the reception of any certificate from a peer entity unless this information is provided by the “Multiple Certificate Status Request” extension.

6.4.2.1.5 The PKI server shall send an OCSP Response if these two conditions are fulfilled:

a) The OCSP Request message is compliant to [RFC-2560] and [RFC-6277]

b) The request contains the information needed by the OCSP server

6.4.2.2 OCSP message specification: OCSP Request

6.4.2.2.1 The OCSP Request shall comply with Table 20, Table 21, Table 22 and Table 23.

OCSPRequest (see RFC-2560)		
tbsRequest	Mandatory	See TBSRequest in Table 21
optionalSignature	Optional	Shall not be used

Table 20: OCSPRequest

TBSRequest (see RFC-2560)		
version	Mandatory	
requestorName	Optional	Shall not be used
requestList	Mandatory	See Request defined in Table 22
requestExtensions	Optional	Shall not be used

Table 21: TBSRequest

Request (see RFC-2560)		
reqCert	Mandatory	See CertID defined in Table 23
singleRequestExtensions	Optional	Shall not be used

Table 22: Request

CertID (see RFC-2560)		
hashAlgorithm	Mandatory	The AlgorithmIdentifier shall be SHA1.
issuerNameHash	Mandatory	
issuerKeyHash	Mandatory	
serialNumber	Mandatory	

Table 23: CertID

6.4.2.3 OCSP message specification: OCSP Response

6.4.2.3.1 The OCSP Response shall comply with Table 24, Table 25, Table 26, Table 27 and Table 28.

OCSPResponse (see RFC-2560)		
responseStatus	Mandatory	
responseBytes	Optional	Shall be used See responseBytes defined in Table 25

Table 24: OCSPResponse

ResponseBytes (see RFC-2560)		
responseType	Mandatory	Shall be id-pkix-ocsp-basic
response	Mandatory	Shall be the DER encoding (see [X.690]) of BasicOCSPResponse defined in Table 26

Table 25: ResponseBytes

BasicOCSPResponse (see RFC-2560)		
tbsResponseData	Mandatory	See ResponseData defined in Table 27
signatureAlgorithm	Mandatory	The AlgorithmIdentifier shall be sha384WithRSAEncryption
signature	Mandatory	
certs	Optional	Shall be used

Table 26: BasicOCSPResponse

ResponseData (see RFC-2560)		
version	Mandatory	
responderID	Mandatory	
producedAt	Mandatory	
responses	Mandatory	See SingleResponse defined in Table 28
responseExtensions	Optional	Shall not be used

Table 27: ResponseData

SingleResponse (see RFC-2560)		
certID	Mandatory	See CertID defined in Table 23
certStatus	Mandatory	
thisUpdate	Mandatory	
nextUpdate	Optional	Shall not be used
singleExtensions	Optional	Shall not be used

Table 28: SingleResponse

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 Addressing

7.2.1.1 A DNS query shall be used to resolve the IP address of KMC, RA and CA.

7.2.1.2 For the KMS entities, the format of the DNS query shall comply to § 9.14.5 of [EIRENE SRS].

7.2.1.3 For RA and CA, the FQDN to be used shall be configured in each KMS entity.

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 an example of how to compute the key database checksum.

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

$KS_{EXAMPLE\_1}$	$KS_{EXAMPLE\_2}$	$KS_{EXAMPLE\_3}$
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**