Document Title	Specification of MACsec Key Agreement
Document Category	Software specification (SWS)
Document UID	1066
Document Status	published
Part of AUTOSAR Standard	Classic Platform
Part of Standard Release	R26-11

Document Change History

Date	Release	Changed by	Description
2026-11-26	R26-11	AUTOSAR Release Management	• Support of CANsec key agreement: new callback functions, the Mka_CanSecConfigType datatype and related configuration parameters • New requirements and constraints for ICV calculation at transmission side and ICV validation at reception side, including handling of the MAC source address via meta data • IdsM security event reporting: IdsM_ReportSecurityEvent added as optional interface, IdsM imported types added, and security event context data definitions revised • Revision of the ECUC configuration model and editorial changes
2025-11-27	R25-11	AUTOSAR Release Management	• Introduction of LSduR for communication path • Configuration of MACsec bypasses based on VLAN and EtherType moved to EthTrcvDrv and EthSwtDrv • New requirements added to detail the usage of the APIs • New interfaces added to both mandatory and optional lists • New sequence diagrams for the interaction of AUTOSAR modules for an MKA's Key Server and Peer node • Fixes in ECUC model

△

			△ • [CP_SWS_Mka_00022] set as not applicable requirement • Editorial changes
2024-11-27	R24-11	AUTOSAR Release Management	• Configurable interface added • Editorial changes • 2 Jobs Reference to Csm Encrypt added ([ECUC_Mka_00074], [ECUC_Mka_00075]) • [CP_SWS_Mka_91019], [CP_SWS_Mka_91024], [CP_SWS_Mka_91031], [CP_SWS_Mka_91032], [CP_SWS_Mka_91033], [CP_SWS_Mka_91001], [CP_SWS_Mka_91014], [CP_SWS_Mka_91008] revised • Description of callback notifications is changed • [CP_SWS_Mka_00303] removed • Security event context data definition tables incorporated • Titles added to TraceItems • CONC 710 (Deterministic Communication with TSN - Concept Part 8) incorporated • Description of return values modified
2023-11-23	R23-11	AUTOSAR Release Management	• MKA Security Events incorporated
2022-11-24	R22-11	AUTOSAR Release Management	• Initial release

Disclaimer

This work (specification and/or software implementation) and the material contained in it, as released by AUTOSAR, is for the purpose of information only. AUTOSAR and the companies that have contributed to it shall not be liable for any use of the work.

The material contained in this work is protected by copyright and other types of intellectual property rights. The commercial exploitation of the material contained in this work requires a license to such intellectual property rights.

This work may be utilized or reproduced without any modification, in any form or by any means, for informational purposes only. For any other purpose, no part of the work may be utilized or reproduced, in any form or by any means, without permission in writing from the publisher.

The work has been developed for automotive applications only. It has neither been developed, nor tested for non-automotive applications.

The word AUTOSAR and the AUTOSAR logo are registered trademarks.

Table of Contents

1 Introduction and functional overview 8

2 Acronyms and Abbreviations 9

3 Related documentation 10
3.1 Input documents & related standards and norms . . . 10
3.2 Related specification . . . 10

4 Constraints and assumptions 11
4.1 Limitations . . . 11
4.2 Applicability to car domains . . . 12

5 Dependencies to other modules 13
5.1 Connection to Linklayer Sdu Routing Module (LSduR) . . . 13
5.2 Connection to Ethernet Interface (EthIf) . . . 13
5.3 Indirect connection to EthDriver, EthSwitchDriver and EthTransceiver-Driver . . . 14
5.4 Connection to Crypto Service Manager (CSM) . . . 14

6 Requirements Tracing 16

7 Functional specification 18
7.1 Background and rationale . . . 18
7.2 Motivation . . . 18
7.2.1 Functional components . . . 20
7.2.1.1 Port Access Entity (PAE) . . . 20
7.2.1.2 MACsec Key Agreement Entity (KaY) . . . 21
7.3 General Requirements . . . 21
7.4 Limitations . . . 24
7.4.1 Limitations on MKA Entity . . . 25
7.5 Cryptographic requirements . . . 25
7.6 MKPDU handling . . . 26
7.7 Communication with MACsec Entity (SecY) . . . 30
7.8 Configurable behavior of MKA . . . 33
7.8.1 MKA behavior . . . 35
7.8.2 MKA Error Handling . . . 37
7.9 Error Classification . . . 38
7.9.1 Development Errors . . . 39
7.9.2 Runtime Errors . . . 39
7.9.3 Production Errors . . . 39
7.9.3.1 MKA_E_TIMEOUT_INSTANCE . . . 39
7.9.3.2 MKA_E_KEY_NOT_PRESENT_INSTANCE . . . 40
7.9.3.3 MKA_E_KEY_MISMATCH_INSTANCE . . . 40
7.9.3.4 MKA_E_ALGO_MISMATCH_INSTANCE . . . 40

7.10 Security Events . . . 41
8 API specification 44
8.1 Imported types . . . 44
8.2 Type definitions . . . 44
8.2.1 Mka_MacSecConfigType . . . 44
8.2.2 Mka_ValidateFramesType . . . 45
8.2.3 Mka_ConfidentialityOffsetType . . . 46
8.2.4 Mka_Stats_Tx_SecYType . . . 46
8.2.5 Mka_Stats_Rx_SecYType . . . 47
8.2.6 Mka_Stats_Tx_ScType . . . 48
8.2.7 Mka_Stats_Rx_ScType . . . 48
8.2.8 Mka_SakKeyPtrType . . . 49
8.2.9 Mka_PermissiveModeType . . . 50
8.2.10 Mka_Stats_SecYType . . . 51
8.2.11 Mka_PaeStatusType . . . 51
8.2.12 Mka_MkaStatusType . . . 52
8.2.13 Mka_ConfigType . . . 52
8.2.14 Mka_CanSecConfigType . . . 53
8.3 Function definitions . . . 53
8.3.1 Mka_Init . . . 54
8.3.2 Mka_SetCknStatus . . . 54
8.3.3 Mka_GetCknStatus . . . 55
8.3.4 Mka_SetEnable . . . 56
8.3.5 Mka_GetEnable . . . 56
8.3.6 Mka_GetPaeStatus . . . 57
8.3.7 Mka_SetPaePermissiveMode . . . 58
8.3.8 Mka_StartPae . . . 58
8.3.9 Mka_GetMacSecStatistics . . . 59
8.3.10 Mka_LinkStateChange . . . 60
8.3.11 Mka_GetMacSecStatisticsNotification . . . 61
8.3.12 Mka_RxIndication . . . 62
8.3.13 Mka_TxConfirmation . . . 62
8.3.14 Mka_MacSecUpdateSecYNotification . . . 63
8.3.15 Mka_MacSecAddTxSaNotification . . . 64
8.3.16 Mka_MacSecAddRxSaNotification . . . 65
8.3.17 Mka_CanSecUpdateSecYNotification . . . 66
8.3.18 Mka_CanSecAddTxSaNotification . . . 66
8.3.19 Mka_CanSecAddRxSaNotification . . . 67
8.3.20 Mka_GetCanSecSecYStatisticsNotification . . . 68
8.3.21 Mka_GetCanSecTxScStatisticsNotification . . . 68
8.3.22 Mka_GetCanSecRxScStatisticsNotification . . . 69
8.3.23 Mka_CanControllerModeIndication . . . 70
8.4 Scheduled functions . . . 70

8.4.1 Mka_MainFunction . . . 70
8.5 Expected interfaces . . . 71
8.5.1 Mandatory interfaces . . . 71
8.5.2 Optional interfaces . . . 72
8.5.3 Configurable interfaces . . . 75
8.6 Service Interfaces . . . 76
9 Sequence diagrams 77
9.1 Communication initialization with MACsec . . . 78
9.2 Communication initialization with MACsec and Switch . . . 79
9.3 MKA exchange - KeyServer . . . 80
9.4 MKA exchange - Peer . . . 81
10 Configuration specification 82
10.1 How to read this chapter . . . 82
10.2 Containers and configuration parameters . . . 82
10.2.1 Mka . . . 82
10.2.2 MkaGeneral . . . 83
10.2.3 MkaPaeConfiguration . . . 87
10.2.4 MkaCryptoAlgoConfig . . . 91
10.2.5 MkaCipherSuites . . . 96
10.2.6 MkaPaeInstance . . . 97
10.2.7 MkaPaeRxPdu . . . 102
10.2.8 MkaPaeTxPdu . . . 103
10.2.9 MkaKay . . . 104
10.2.10 MkaKayParticipant . . . 107
10.2.11 MkaKayDemEventParameterRefs . . . 114
10.2.12 MkaSecurityEventRefs . . . 116
10.3 Published Information . . . 119
A Not applicable requirements 120
B Change history of AUTOSAR traceable items 121
B.1 Traceable item history of this document according to AUTOSAR Release R23-11 . . . 121
B.1.1 Added Specification Items in R23-11 . . . 121
B.1.2 Changed Specification Items in R23-11 . . . 121
B.1.3 Deleted Specification Items in R23-11 . . . 122
B.1.4 Added Constraints in R23-11 . . . 122
B.1.5 Changed Constraints in R23-11 . . . 122
B.1.6 Deleted Constraints in R23-11 . . . 122
B.2 Traceable item history of this document according to AUTOSAR Release R24-11 . . . 123
B.2.1 Added Constraints in R24-11 . . . 123
B.2.2 Changed Constraints in R24-11 . . . 123
B.2.3 Deleted Constraints in R24-11 . . . 123

B.2.4 Added Specification Items in R24-11 . . . 123
B.2.5 Changed Specification Items in R24-11 . . . 130
B.2.6 Deleted Specification Items in R24-11 . . . 130
B.3 Traceable item history of this document according to AUTOSAR Release R25-11 . . . 130
B.3.1 Added Constraints in R25-11 . . . 130
B.3.2 Changed Constraints in R25-11 . . . 130
B.3.3 Deleted Constraints in R25-11 . . . 130
B.3.4 Added Specification Items in R25-11 . . . 131
B.3.5 Changed Specification Items in R25-11 . . . 131
B.3.6 Deleted Specification Items in R25-11 . . . 132
B.4 Traceable item history of this document according to AUTOSAR Release R26-11 . . . 133
B.4.1 Added Constraints in R26-11 . . . 133
B.4.2 Changed Constraints in R26-11 . . . 133
B.4.3 Deleted Constraints in R26-11 . . . 133
B.4.4 Added Specification Items in R26-11 . . . 133
B.4.5 Changed Specification Items in R26-11 . . . 134
B.4.6 Deleted Specification Items in R26-11 . . . 137

1 Introduction and functional overview

This specification describes the functionality, API and the configuration for the AUTOSAR Basic Software module MKA.

2 Acronyms and Abbreviations

The glossary below includes acronyms and abbreviations relevant to the MKA module that are not included in the [1, AUTOSAR glossary].

Abbreviation / Acronym:	Description:
AN	Association Number
CA	Secure Connectivity Association
CAK	Secure Connectivity Association Key
DA	Destination Address
ICV	Integrity Check Value
KaY	MAC Security Key Agreement Entity
MACsec	Media Access Control Security
MKA	MACsec Key Agreement protocol (IEEE Std 802.1X)
MKPDU	MACsec Key Agreement Protocol Data Unit
MPDU	MACsec Protocol Data Unit
PAE	Port Access Entity
PN	Packet Number
SA	Secure Association or Source Address, as applicable
SAI	Secure Association Identifier
SAK	Secure Association Key
SC	Secure Channel
SCI	Secure Channel Identifier
SecTAG	MAC Security TAG
SecY	MAC Security Entity
SL	Short Length
SSCI	Short Secure Channel Identifier
TLV	Tag-Length-Value

Table 2.1: Acronyms and abbreviations used in the scope of this Document

3 Related documentation

3.1 Input documents & related standards and norms

[1] Glossary
AUTOSAR_FO_TR_Glossary

[2] General Specification of Basic Software Modules
AUTOSAR_CP_SWS_BSWGeneral

[3] Requirements on MACsec
AUTOSAR_FO_RS_MACsec

[4] IEEE Standard for Local and metropolitan area networks-Media Access Control (MAC) Security
https://ieeexplore.ieee.org/document/8585421

[5] IEEE Standard for Local and Metropolitan Area Networks–Port-Based Network Access Control
https://ieeexplore.ieee.org/document/9018454

[6] Advanced Encryption Standard (AES) Key Wrap Algorithm
https://tools.ietf.org/html/rfc3394

[7] Specification of Ethernet Interface
AUTOSAR_CP_SWS_EthernetInterface

[8] System Template
AUTOSAR_CP_TPS_SystemTemplate

3.2 Related specification

AUTOSAR provides

- a General Specification on Basic Software modules [2, SWS BSW General], which is also valid for MKA.
- a MACsec Requirements Specification [3, RS MACsec] which is also valid for MKA.

Thus, the specification [2, SWS BSW General] shall be considered as additional and required specification for MKA.

4 Constraints and assumptions

4.1 Limitations

This document does not cover the integration neither requirements of the MACsec protocol as it is an IEEE published standard [4, IEEE 802.1AE-2018].

The AUTOSAR MACsec implementation currently has the following limitations:

- Only participants authentication based on Connectivity Association pre-shared keys (CAKs) is supported. (EAP-TLS, EAP-IKEv2, and other variants are not supported).
- Only MACsec between direct peers is supported (e.g. Point-to-Point configurations).
- Point-to-Multipoint configurations are not supported.
- In-service upgrades with EAPoL-MKA frames are not supported.
- Temporary suspension of MKA operation is not supported.
- MACsec Cipher Suites is the only currently supported EAPoL-Announcement TLV.
  The following EAPoL Announcements are currently not required:
  - Access Information → TLV Type 111
  - Key Management Domain → TLV Type 113
  - NID → TLV Type 114
- Dynamic Key Server election based on Key Server priority is not supported (Roles are set per configuration and fixed).
- The following MKPDU Parameter sets are currently not required:
  - Distributed CAK → Parameter set type 5
  - KMD → Parameter set type 6
  - ICV Indicator → Parameter set type 255

For the AUTOSAR CANsec implementation, it is planned that the MKA module includes the CANsec key agreement entity (KaY) once the specification has been published by CiA. Until then, the MKA module already contains the APIs for the control plane API-flow. These APIs can stay unimplemented until the functional specification is added in a future release. A respective note has been added to the affected APIs.

4.2 Applicability to car domains

Automotive systems require quicker start-up times for the devices connected to the on-board network, hence the protocol convergence time must be tuned accordingly.

5 Dependencies to other modules

The MACsec Key Agreement (MKA) Module has interfaces with the following modules:

1. EthIf → To configure, control, and monitor the MACsec Entity (per SW or HW).
2. CSM:
   - Protect outgoing MKA messages and validate incoming MKA messages.
   - Generate, encrypt, and decrypt session keys (SAKs).
3. LSduR:
   - forward MKPDUs from the MKA module to the EthIf
   - forward MKPDUs from the EthIf to the MKA module

5.1 Connection to Linklayer Sdu Routing Module (LSduR)

The MKA module and the LSduR are connected in order to route transmission and reception MKA messages.

5.2 Connection to Ethernet Interface (EthIf)

The MKA module and the EthIf are connected in order to:

- Provide MACsec specific parameters to the lower layers.
- Orchestrate the Link-Up and Link-Down signaling of the interfaces for upper layers (i.e. through the EthSM).

In case an Ethernet Interface is MACsec protected, it will use a specific MKA module instance to configure the MACsec Entities (HW or SW) for transmission and reception.

In case the MACsec protected Ethernet Interface is required to be ACTIVE by the EthSM, after the signaling of the physical Link-up from the specific transceiver or Switch port(*ETH_MODE_ACTIVE*), the EthIf will delegate the establishment of at least one Secure Channel with the communication peers, to the referred MKA instance.
Once the MACsec Secure Channel is established and both participants can successfully receive and transmit, the Ethernet Interface will signal the Link-Up to the upper layers (e.g. through the Ethernet State Manager).

During the lifetime of the established SCs, the MKA module will maintain them alive by communicating with the MACsec Entities through the Ethernet Interface module. That means, updating the SC specific parameters in the MACsec Entities (Phy, Switch, or

SW Entity).

Detailed information: The trigger to the MKA module to start the MACsec SC establishment is done after the EthTrcv or EthSwt mode indication to ACTIVE and before indicating this state to the EthSM (that means, the EthSM will stay in the *ETHSM_STATE_WAIT_TRCVLINK* state, as in this state the EthSM and the underlying EthIf is starting up the physical network interface, but the upper layer protocols (e.g., in TCP/IP) are not started yet).
Once triggered, the MKA module can start the needed actions to establish a MACsec Secure Channel through the provided port. If MACsec is not configured in the port, the MKA module call will be skipped.

5.3 Indirect connection to EthDriver, EthSwitchDriver and EthTransceiverDriver

In case the MACsec Entity is offloaded to a HW device, the MKA module is indirectly connected to the EthDriver, EthSwitchDriver, and EthTranceiverDriver through the EthInterface. This connection is needed in order to establish, configure, and manage the needed MACsec Secure Channels. There are functions in the interface of the EthDriver, EthSwitchDriver and EthTransceiverDriver for that purpose.

Establishing a Secure Channel is done via the MACsec Key Agreement protocol, the MKA module will handle all protocol steps. These specific protocol datagrams are setup and organized by the MKA module. Thus, the MKA module provides via the existing function call the datagram to the Ethernet Interface, which then sends the datagram to the communication peer. This behavior is handled via a specific pair EtherType and message type, and is set via the interface. With this EtherType, the Ethernet Interface will handle the datagram on Rx and Tx trace.

5.4 Connection to Crypto Service Manager (CSM)

The MKA module requires a connection to the cryptographic BSW modules of AUTOSAR. This allows the MKA module to derive and use the needed keys and to interact with the cryptographic algorithms as specified in [5, IEEE-802.1X-2020] and [4, IEEE-802.1AE-2018].

For cryptographic usage, the MKA module needs following support from the BSW crypto:

- KDF (as described in [5, IEEE-802.1AE-2018] chapter 6.2.1) to derive ICK and KEK from CAK.
- AES-CMAC, which uses AES CMAC with 128bits, using ICK to generate and validate MKA message ICVs.

- A function to generate random data (for SAK and Member Identifier).
- AES-KEYWRAP based on [6, RFC 3394] to encrypt keys for transmission.

6 Requirements Tracing

The following tables reference the requirements specified in the documents listed in Chapter 3.2 and links to the fulfillment of these. Please note that if column “Satisfied by” is empty for a specific requirement this means that this requirement is not fulfilled by this document.

Requirement	Description	Satisfied by
[FO_RS_MACsec_-00001]	MACsec Protocol support	[CP_SWS_Mka_00037] [CP_SWS_Mka_00999]
[FO_RS_MACsec_-00002]	MACsec Key Agreement Protocol support	[CP_SWS_Mka_00001] [CP_SWS_Mka_00002] [CP_SWS_Mka_00005] [CP_SWS_Mka_00006] [CP_SWS_Mka_00007] [CP_SWS_Mka_00008] [CP_SWS_Mka_00009] [CP_SWS_Mka_00011] [CP_SWS_Mka_00015] [CP_SWS_Mka_00016] [CP_SWS_Mka_00017] [CP_SWS_Mka_00024] [CP_SWS_Mka_00031] [CP_SWS_Mka_00032] [CP_SWS_Mka_00037] [CP_SWS_Mka_00038] [CP_SWS_Mka_00039] [CP_SWS_Mka_00040] [CP_SWS_Mka_00041] [CP_SWS_Mka_00042] [CP_SWS_Mka_00043] [CP_SWS_Mka_00044] [CP_SWS_Mka_00045] [CP_SWS_Mka_00047] [CP_SWS_Mka_00048] [CP_SWS_Mka_00049] [CP_SWS_Mka_00050] [CP_SWS_Mka_00051] [CP_SWS_Mka_CONSTR_00021] [CP_SWS_Mka_CONSTR_00022] [CP_SWS_Mka_CONSTR_00023] [CP_SWS_Mka_CONSTR_00024]
[FO_RS_MACsec_-00003]	Using MACsec on external communication links	[CP_SWS_Mka_00001] [CP_SWS_Mka_00002] [CP_SWS_Mka_00006] [CP_SWS_Mka_00008] [CP_SWS_Mka_00011] [CP_SWS_Mka_00024]
[FO_RS_MACsec_-00004]	Configure which Ethernet ports use MACsec	[CP_SWS_Mka_00002]
[FO_RS_MACsec_-00005]	MACsec status control	[CP_SWS_Mka_00026] [CP_SWS_Mka_00027] [CP_SWS_Mka_00028] [CP_SWS_Mka_00029] [CP_SWS_Mka_00030] [CP_SWS_Mka_00036]
[FO_RS_MACsec_-00006]	MACsec support for Adaptive AUTOSAR Platform	[CP_SWS_Mka_00036]
[FO_RS_MACsec_-00008]	Configuration of unprotected traffic (for Hardware-based MACsec)	[CP_SWS_Mka_00038] [CP_SWS_Mka_00039] [CP_SWS_Mka_00040] [CP_SWS_Mka_00041] [CP_SWS_Mka_00042] [CP_SWS_Mka_00043] [CP_SWS_Mka_00044] [CP_SWS_Mka_00045] [CP_SWS_Mka_00050] [CP_SWS_Mka_00051] [CP_SWS_Mka_CONSTR_00021] [CP_SWS_Mka_CONSTR_00023] [CP_SWS_Mka_CONSTR_00024]
[FO_RS_MACsec_-00009]	MACsec Security Events	[CP_SWS_Mka_00025] [CP_SWS_Mka_00301] [CP_SWS_Mka_00304] [CP_SWS_Mka_00305] [CP_SWS_Mka_00306] [CP_SWS_Mka_00307] [CP_SWS_Mka_00308]
[FO_RS_MACsec_-00010]	Support of integrity and confidentiality	[CP_SWS_Mka_00008]
[FO_RS_MACsec_-00011]	MAC Security TAG	[CP_SWS_Mka_00999]
[FO_RS_MACsec_-00012]	MACsec EtherType	[CP_SWS_Mka_00999]
[FO_RS_MACsec_-00017]	Support of Extended Packet Number (XPN)	[CP_SWS_Mka_00008]

▽

△

Requirement	Description	Satisfied by
[FO_RS_MACsec_-00018]	Secure Channel Identifier (SCI)	[CP_SWS_Mka_00999]
[FO_RS_MACsec_-00019]	Secure Data	[CP_SWS_Mka_00999]
[FO_RS_MACsec_-00020]	Integrity Check Value (ICV)	[CP_SWS_Mka_00034]
[FO_RS_MACsec_-00021]	Protect function in software solution	[CP_SWS_Mka_00999]
[FO_RS_MACsec_-00022]	Validation function in software solution	[CP_SWS_Mka_00999]
[FO_RS_MACsec_-00023]	Support of MKA Packets	[CP_SWS_Mka_00001] [CP_SWS_Mka_00008]
[FO_RS_MACsec_-00024]	Pre-shared key support	[CP_SWS_Mka_00001] [CP_SWS_Mka_00005] [CP_SWS_Mka_00016] [CP_SWS_Mka_00033]
[FO_RS_MACsec_-00025]	Key selection via CKN	[CP_SWS_Mka_00001] [CP_SWS_Mka_00005] [CP_SWS_Mka_00006]
[FO_RS_MACsec_-00027]	Support of AES ciphers with at least 128 bits of key length	[CP_SWS_Mka_00009]
[FO_RS_MACsec_-00028]	Support of AES ciphers with 256 bits of key length	[CP_SWS_Mka_00009]
[FO_RS_MACsec_-00029]	Support of Key Encryption Key (KEK)	[CP_SWS_Mka_00001] [CP_SWS_Mka_00006] [CP_SWS_Mka_00022]
[FO_RS_MACsec_-00030]	Support of Integrity Check Value Key (ICK)	[CP_SWS_Mka_00006] [CP_SWS_Mka_00022]
[FO_RS_MACsec_-00031]	Support of Key Derivation Function (KDF)	[CP_SWS_Mka_00007]
[FO_RS_MACsec_-00032]	List of minimal supported cipher suites	[CP_SWS_Mka_00009] [CP_SWS_Mka_00035]
[FO_RS_MACsec_-00033]	Validation function for ICVs	[CP_SWS_Mka_00001] [CP_SWS_Mka_00006] [CP_SWS_Mka_00034]
[FO_RS_MACsec_-00034]	Generation function for ICVs	[CP_SWS_Mka_00999]
[FO_RS_MACsec_-00035]	Key Handling with combined HSM and MACsec functionality	[CP_SWS_Mka_00999]
[FO_RS_MACsec_-00036]	Interframe gap configuration of Ethernet controller	[CP_SWS_Mka_00999]
[FO_RS_MACsec_-00037]	MACsec participants per link	[CP_SWS_Mka_00015]
[FO_RS_MACsec_-00038]	MKA SC establishment retry phase	[CP_SWS_Mka_00012]
[FO_RS_MACsec_-00039]	MKA rekey conditions	[CP_SWS_Mka_00013] [CP_SWS_Mka_00014]
[SRS_BSW_00323]	All AUTOSAR Basic Software Modules shall check passed API parameters for validity	[CP_SWS_Mka_91035]
[SRS_BSW_00337]	Classification of development errors	[CP_SWS_Mka_00200] [CP_SWS_Mka_00201] [CP_SWS_Mka_00202] [CP_SWS_Mka_00203] [CP_SWS_Mka_91035]
[SRS_BSW_00385]	List possible error notifications	[CP_SWS_Mka_00200] [CP_SWS_Mka_00201] [CP_SWS_Mka_00202] [CP_SWS_Mka_00203] [CP_SWS_Mka_91035]

Table 6.1: Requirements Tracing

7 Functional specification

7.1 Background and rationale

A detailed description of the MACsec and MACsec Key Agreement protocols is included in [3] Chapter 4.1 *"Functional Overview"*.

7.2 Motivation

The aim of this document is to specify how to integrate the MKA Module in the Software Layered Architecture of the AUTOSAR Classic Platform.

The purpose of the MACsec Key Agreement Module is to provide a method for discovering MACsec peers and negotiate the security keys needed to secure the link. The MKA Module is responsible for:

- Generating (outgoing) and processing (incoming) MKPDUs.
- Identify and authenticate other partners belonging to the same Connectivity Association (CA).
- Configure and supply the parameters and cryptographic data to the MACsec Entity (per SW or HW) for the respective Secure Channel and Secure Associations established.
- Keep the established Secure Channel (SC) and Secure Association (SA) information updated.
- Refresh keys for an specific Secure Channel to allow exchanging Secure Association Keys (SAKs) without disrupting the communication channel.

The MKA module supports:

- The configuration, initialization, and maintenance of Port Access Entities (PAEs).
- The configuration, initialization, shutdown, and maintenance of MKA Entities (KaYs) which belong to an specific PAE.
- The communication with other AUTOSAR Modules to initialize, update, and shutdown MACsec related parameters into the MACsec Entity (SecY).

The limitations of the referred IEEE standards are included in Chapter 4.

Figure 7.1 depicts the MACsec Key Agreement protocol parameter sets exchanged to establish a Secure Channel and finally enable MACsec protected secure communication.

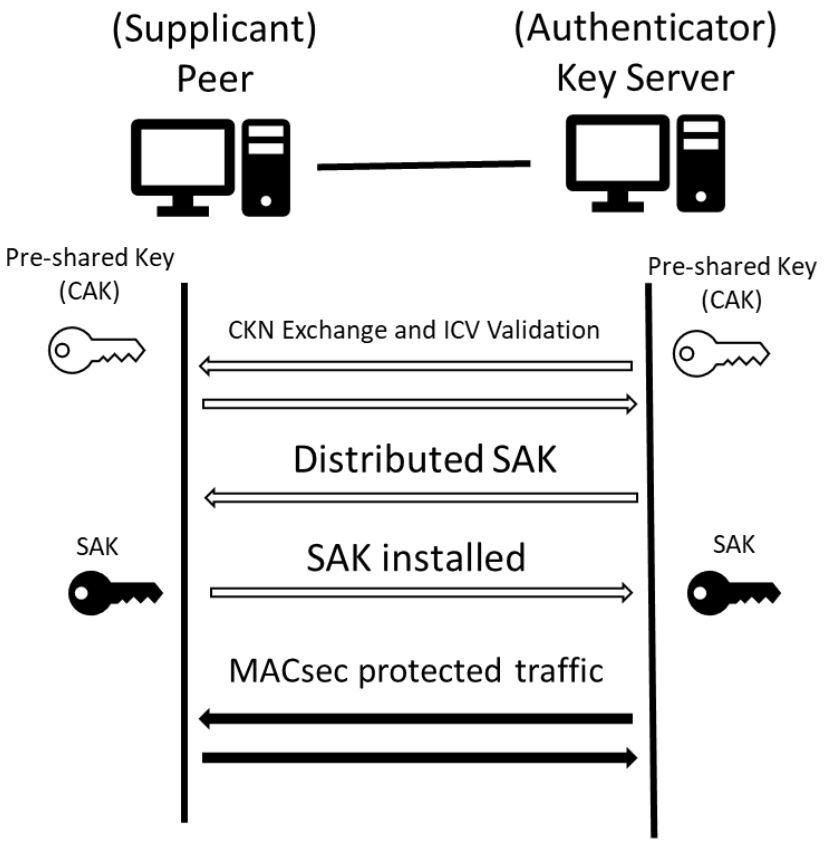

Figure 7.1: fig: MACsec Key Agreement sequence with pre-shared key

Figure 7.2 provides an architecture overview of the AUTOSAR MKA module in the Layered Software Architecture.

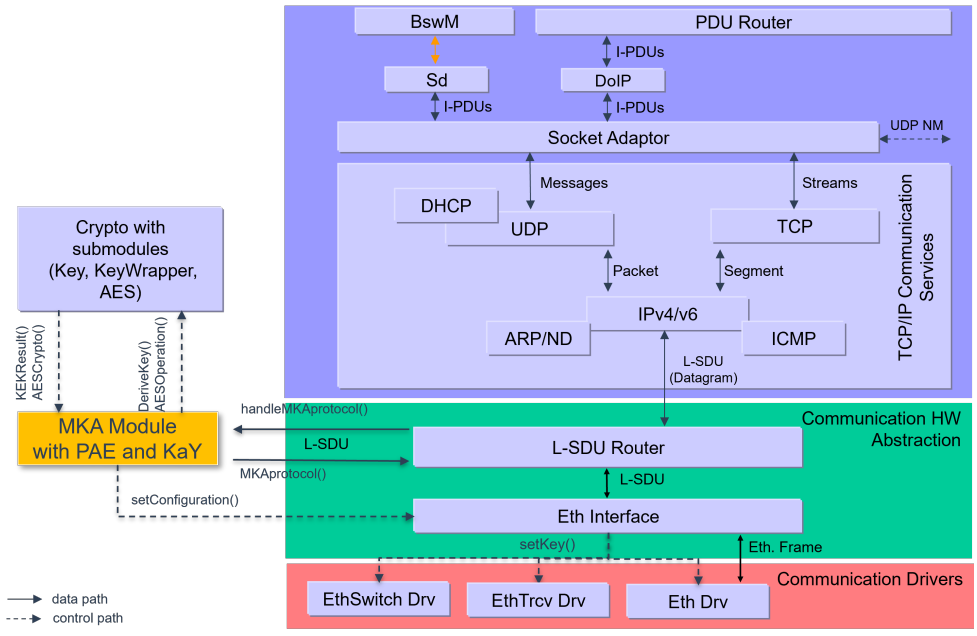

Figure 7.2: MKA module in the SW Architecture of AUTOSAR CP

Based on the IEEE standards ([4, IEEE-802.1AE-2018] and [5, IEEE-802.1X-2020]), the system allows to configure, setup, and run integrity protected and/or encrypted data communication per Ethernet port. This implies that the architecture, specification, and the later implementation enables MACsec on each port when this is configured via AUTOSAR configuration.
Additionally, MACsec allows to "bypass" traffic based on EtherType, VLAN-ID or destination MAC address. This means that MACsec handling could pass selected traffic unprotected based on the configured bypass rules.
MACsec processing is statically configured in advance for the entities supporting the protocol, the configuration is based on rules. This includes rules for determining the bypassed traffic. Since the bypassed traffic could be based on VLAN-IDs, the handling of MACsec protected networks interacts with VLAN-based communication. Bypassed traffic is available as soon as a link-up of the used Ethernet hardware (e.g. PHY, Ethernet switch port, Ethernet controller) is signaled, while protected traffic needs to wait until MACsec and its Key-Agreement protocol has established a secured connection.

The Ethernet Interface will behave different for protected and unprotected traffic. The Ethernet Interface (EthIf) sequence is modified in case the controller (and therefore EthSwt and/or EthTrcv) has to deal with a MACsec protected port.

After receiving a Link-up indication from a MACsec protected port (which could be after a Switch), the Ethernet Interface will trigger the MKA Module to start the MKA Sequence for the corresponding port. Once the MKA module signals the success of the MKA sequence and therefore the proper configuration of MACsec on the port, the Link-Up is propagated to the upper layers (i.e., EthSM or others through the corresponding UL_LinkStateChg method). This is essential to start the upper layer protocols (e.g., SOME/IP-SD) as soon as the MACsec protected link is ready to be used.

This applies as well in case Groups of Ports are defined for the respective network.

7.2.1 Functional components

7.2.1.1 Port Access Entity (PAE)

In the [5, IEEE-802.1X-2020] standard the Port Access Entity (PAE) describes the (SW) entity that controls an Ethernet port. This includes allowing traffic to flow or not to flow but also controlling the MACsec functionality based on the MACsec Key Agreement protocol (MKA).

[5, IEEE-802.1X-2020] defines port based authentication over EAP and, as a particular use case, the MACsec Key Agreement protocol over EAP. In the current version of this document, authentication over EAP is not supported and therefore only authentication based on pre-shared CA Keys (CAKs) is relevant.

The PAE will take care of orchestrating the initialization and shutdown of MKA instances (defined in next section) and setting the status (enabled/disabled) of the physical or virtual controlled port based on the feedback provided by the underlying KaYs.

For a better understanding of the PAE structure, refer to [5, IEEE-802.1X-2020] chapter 12.

7.2.1.2 MACsec Key Agreement Entity (KaY)

Each PAE can have one or multiple MACsec Key Agreement participants (KaY participants) depending on the CKNs assigned to the Port Access Entity.
Each KaY is responsible of recognizing peers which belong to the same CA, distribute and/or install SAKs, and keep the MACsec information up to date during the SC lifetime.
The KaY handles the MKA protocol behavior, including the generation and process of MKPDUs, the control of the cipher suites to use and, the maintenance of the MACsec related parameters of the MACsec Entity (SecY), including Keys (SAKs).

7.3 General Requirements

[CP_SWS_Mka_00001]

Status:	DRAFT
Upstream requirements:	FO_RS_MACsec_00002, FO_RS_MACsec_00003, FO_RS_MACsec_00023, FO_RS_MACsec_00024, FO_RS_MACsec_00025, FO_RS_MACsec_00029, FO_RS_MACsec_00033

⌈The MKA Module shall implement the EAP-MKA protocol version 3 as specified in [5, IEEE-802.1X-2020] chapter 9 and AUTOSAR Foundation [3, RS_MACsec].⌋

Note: The MKA Module should be modeled as described in [5, IEEE-802.1X-2020] chapter 12.
For the excluded parts please refer to Chapter 4.1.

[CP_SWS_Mka_00002]

Status:	DRAFT
Upstream requirements:	FO_RS_MACsec_00002, FO_RS_MACsec_00003, FO_RS_MACsec_00004

⌈The MKA Module shall support 1 to n independent Port Access Entities (PAEs) running at the same time through different ports.⌋

Note: Each physical (Switch port or transceiver) or virtual (MACsec per SW) port will support 0 (No MACsec) or 1 PAE.

[CP_SWS_Mka_00005]

Status: DRAFT
Upstream requirements: FO_RS_MACsec_00002, FO_RS_MACsec_00024, FO_RS_MACsec_00025

⌈The MKA Module shall support configuring 1 to n CKNs in an specific Port Access Entity (PAE).
Each configured CKN will start a parallel MACsec participant entity (i.e. `MkaKayParticipant`) through the mentioned PAE.
Repeated CKNs shall be treated as one for an specific PAE.⌋

Note: It is recommended to implement a configuration check to avoid duplicated CKNs referred under the same `MkaKay` instance.

[CP_SWS_Mka_00006]

Status: DRAFT
Upstream requirements: FO_RS_MACsec_00002, FO_RS_MACsec_00003, FO_RS_MACsec_00025, FO_RS_MACsec_00029, FO_RS_MACsec_00030, FO_RS_MACsec_00033

⌈An MKA KaY participant (`MkaKayParticipant`) shall not start transmitting or processing MKPDUs until its respective CAK is available and the derived keys (ICK and KEK) are ready.⌋

[CP_SWS_Mka_00007]

Status: DRAFT
Upstream requirements: FO_RS_MACsec_00002, FO_RS_MACsec_00031

⌈The MKA Module shall support generation of SAKs based on:

- Key Derivation Function (KDF), see [5, IEEE-802.1X-2020] chapter 9.8.1.
- Random Number Generator (RNG)

⌋

[CP_SWS_Mka_00008]

Status: DRAFT
Upstream requirements: FO_RS_MACsec_00002, FO_RS_MACsec_00003, FO_RS_MACsec_00010, FO_RS_MACsec_00017, FO_RS_MACsec_00023

⌈The MKA Module shall support the following MKPDU Parameter sets:

- Basic Parameter Set
- Live Peer List → Parameter set type 1
- Potential Peer List → Parameter set type 2
- MACsec SAK Use → Parameter set type 3
- Distributed SAK → Parameter set type 4
- Announcement → Parameter set type 7

- XPN → Parameter set type 8

⌋

[CP_SWS_Mka_00009]

Status: DRAFT
Upstream requirements: FO_RS_MACsec_00002, FO_RS_MACsec_00027, FO_RS_MACsec_00028, FO_RS_MACsec_00032

⌈The MKA Module shall implement the EAPoL-MKA-Announcement with TLV type 112 (MACsec cipher suites) as specified in [5, IEEE-802.1X-2020] chapter 11.12.3.⌋

Note: The MACsec cipher suite announcement serves for the Key Server to recognize the ciphers supported by the other end.
The EAPoL-Announcement TLV shall be transmitted as a parameter on an EAPoL-MKA Announcement Parameter Set as defined in Figure 11-15 of [5, IEEE-802.1X-2020].

[CP_SWS_Mka_00011]

Status: DRAFT
Upstream requirements: FO_RS_MACsec_00002, FO_RS_MACsec_00003

⌈The role of an MKA instance (`MkaKay`) shall be set per configuration (i.e. `MKA_KEY_SERVER` or `MKA_PEER`) (`MkaRole`).
The Key Server priority shall be configurable (`MkaKeyServerPriority`), in case it is not specifically provided in configuration the following values shall be used:

- Key Server = 0
- Peer = 255

⌋

[CP_SWS_Mka_00012]

Status: DRAFT
Upstream requirements: FO_RS_MACsec_00038

⌈The MKA Module shall support retry for the MKA sequence.
If an MKA KaY participant (`MkaKayParticipant`) cannot successfully identify or successfully establish a SC with any participant in the link , it should retry the MKA sequence following a per configuration parametrized *retry base delay with Exponential Back-off* (`MkaRetryBaseDelay`) until a *retry cyclic delay* (`MkaRetryCyclicDelay`).⌋

Note: As an example, in case `MkaRetryBaseDelay` = 0.02 and `MkaRetryCyclicDelay` = 0.5, the retry sequence will be as follows 20ms, 40ms, 80ms, 160ms, 320ms, 500ms, 500ms, ...

[CP_SWS_Mka_00013]
Status: DRAFT
Upstream requirements: FO_RS_MACsec_00039

⌈The MKA Instances (MkaKay) configured with the Key Server (MKA_KEY_SERVER) role shall support re-keying distributed SAKs after a configurable time span (MkaSakRekeyTimeSpan).⌋

Note: The time span is set per configuration.

[CP_SWS_Mka_00014]
Status: DRAFT
Upstream requirements: FO_RS_MACsec_00039

⌈The MKA Instances configured with the Key Server (MKA_KEY_SERVER) role shall rekey distributed SAKs in case the packet number space of one direction (sending or receiving) exceeds:

- 0xC000 0000 for 32-bit PNs.
- 0xC000 0000 0000 0000 for 64-bit PNs (XPN mode).

⌋

Note: This is required in the [5, IEEE-802.1X-2020] standard, chapter 9.8.

[CP_SWS_Mka_00015]
Status: DRAFT
Upstream requirements: FO_RS_MACsec_00002, FO_RS_MACsec_00037

⌈Each MKA instance shall assume exactly two participants per link. Therefore, having exactly one peer.⌋

Note: This implies, that one must take the Key Server role and the other the peer role. This requirement permits a MKA instance to immediately continue with the MKA sequence after detecting and successfully authenticating another participant in the link which belongs to the same CA, avoiding start-up delays.

7.4 Limitations

An overview of non-supported features can be found in Chapter 4.

7.4.1 Limitations on MKA Entity

[CP_SWS_Mka_00016]
Status: DRAFT
Upstream requirements: FO_RS_MACsec_00002, FO_RS_MACsec_00024

⌈The MKA Module may support authentication based on EAP as detailed in [5, IEEE-802.1X-2020] chapter 8.⌋

Note: Authentication based on EAP is not required as the mutual authentication of participants is achieved based on pre-shared Keys.

[CP_SWS_Mka_00017]
Status: DRAFT
Upstream requirements: FO_RS_MACsec_00002

⌈The MKA Module shall support only one MKA Participant (`MkaKayParticipant`) to success per port (i.e. per PAE `MkaPaeInstance`).
If one KaY participant is able to correctly establish a SC, the other started participants (`MkaKayParticipant`) of the same PAE shall give up.⌋

Note: As specified in [CP_SWS_Mka_00005], a PAE instance can initiate 1 to n MKA participant instances (`MkaKayParticipant`) but only one of them shall succeed configuring a Secure Channel in the port (Point-to-Multipoint architecture is not supported).

7.5 Cryptographic requirements

[CP_SWS_Mka_CONSTR_00019]
Status: DRAFT

⌈The MKA Module shall support the following Cipher suites to be configured in the MACsec Entity (either per SW or HW):

- GCM-AES-128
- GCM-AES-256
- GCM-AES-XPN-128
- GCM-AES-XPN-256

⌋

Note: The MKA Module shall support announcing and configuring the listed ciphers (`MkaMacSecCipherSuite`).
Detailed information can be found in [5, IEEE-802.1X-2020] Figure 11-12 and [4, IEEE-802.1AE-2018] chapter 14.3.

[CP_SWS_Mka_CONSTR_00020]

Status: DRAFT

⌈The MKA Module shall support configuring 1 to 4 cipher suites per MkaCryptoAlgoConfig, each of them with an unique MkaMacSecCipherSuitePrio.
The MkaMacSecCipherSuitePrio shall accept the value 1 to 4, being 1 the higher priority and 4 the lowest priority.⌋

Note: The MkaMacSecCipherSuitePrio parameter shall be used by a MkaKayParticipant with MkaRole = MKA_KEY_SERVER to select the cipher suite to use for MACsec with the other participant within the common cipher suites supported (shared with the EAPoL-MKA-Announcement "MACsec cipher suites").

The cryptographic operations like the derivation of MACsec keys and authentication based on CAK pre-shared keys should be delegated to the CSM Module.

Note: For detailed information, refer to [5, IEEE-802.1X-2020] chapter 9.3.

[CP_SWS_Mka_00022]

Status: DRAFT
Upstream requirements: FO_RS_MACsec_00029, FO_RS_MACsec_00030

⌈Derived keys (specifically ICKs and KEKs) may get pre-calculated and stored in tamper proof manner to optimize the initialization time of the module.
SAKs are implicitly excluded from this requirement. SAKs must not be pre-calculated neither reused.⌋

7.6 MKPDU handling

The MKA module need to process and to generate MKPDUs to communicate with its direct link partner. Generated MKPDUs are used for outgoing traffic and processed MKPDUs are handled for incoming traffic. Within the AUTOSAR communication stack, MKPDUs are transmitted and received via PDUs. The MKA module has per MkaPaeInstance one MkaPaeRxPdu and one MkaPaeTxPdu configured. The MkaPaeTxPdu is used to transmit and the MkaPaeRxPdu is used to receive MKPDUs. The PDUs are routed via the LSduR to the EthIf module and vice versa. MKPDUs are transferred on the data line (either with or without VLAN tag) as traffic that is not MACsec protected, i.e. bypassing the MACsec operation of the protected Ethernet port (e.g. PHY). Therefore, the configuration of the LSduR and EthIf need to consider that the MkaPaeRxPdus and the MkaPaeTxPdus are connected to PDUs via the LSduR that refer to an EthIfController where the configuration of the corresponding lower layer driver (e.g. EthTrcv) has a bypass configuration (e.g. EthTrcvMacSecBypassEtherType), i.e. this EthIfController allow MACsec unprotected traffic.

[CP_SWS_Mka_CONSTR_00021] `MkaPaeRxPdu` and `MkaPaeTxPdu` shall refer to an `EthIfController` that allows MACsec unprotected traffic

Status: DRAFT
Upstream requirements: FO_RS_MACsec_00002, FO_RS_MACsec_00008

⌈A `MkaPaeRxPdu` and a `MkaPaeTxPdu` of a specific `MkaPaeInstance` shall refer to PDUs of an `EthIfController` where the associated MACsec protected Ethernet port (e.g. PHY or Ethernet controller or Ethernet switch port) has either MACsec protection disabled or an EtherType bypass rule (e.g. `EthTrcvMacSecBypassEtherType`) configured with value set to the EAPoL Ethernet Type specified by [5, IEEE-802.1X-2020] and the corresponding EthIf PDU configuration fulfills the following constraints:

- the `MkaPaeRxPdu` refers to a `EthIfFrameRxPool` and the `MkaPaeTxPdu` refers to a `EthIfFrameTxPool`, where both PDU pools belong to the same `EthIfFrameConfig` that has `EthIfFrameType` set to the value of the EAPoL Ethernet Type specified by [5, IEEE-802.1X-2020]
- those PDU pools reference this `EthIfController`

⌋

[CP_SWS_Mka_00038] MAC source / destination address of a MKPDU to be transmitted

Status: DRAFT
Upstream requirements: FO_RS_MACsec_00002, FO_RS_MACsec_00008

⌈If the MKA module need to generate a MKPDU for transmission of a specific `MkaPaeInstance`, then the MKA module shall consider at least the following points:

- the MKA shall retain the MAC source address of the Ethernet controller that belongs to this `MkaPaeInstance` by calling `EthIf_GetPhysAddr` with `CtrlIdx` set to a `EthIfController` that references the `EthIfPhysController` which addresses this Ethernet controller
- the MKA shall set the value of the meta data item `ETHERNET_MAC_64` to the configured `MkaDstMacAddress` of the `MkaKay` that belongs to this `MkaPaeInstance`

⌋

[CP_SWS_Mka_CONSTR_00023] Provide MAC destination address of a MKPDU via meta data at transmission side

Status: DRAFT
Upstream requirements: FO_RS_MACsec_00002, FO_RS_MACsec_00008

⌈At each `MkaPaeInstance` the `MkaPaeTxPdu` shall have the meta data item `ETHERNET_MAC_64` configured.⌋

MKPDUs carrying a so-called "integrity check value" (ICV), which is calculated on transmission side and validated on reception side. The value is calculated by considering MAC destination address, MAC source address, EtherType and the MKPDU

without ICV. Please note, at transmission as well as on reception side, the configured MkaDstMacAddress of the corresponding MkaPaeInstance is used for the ICV calculation / validation. The MKA module expects the MKPDU to be received with the configured MkaDstMacAddress. An ICV validation at the MKA module would fail if the MAC destination address of the received Ethernet frame deviates from the configured MkaDstMacAddress, even though the ICV value may be valid with regards to the received Ethernet frame MAC destination address. The system design needs to ensure, that the configured MAC destination address is properly distributed at the affected ECUs.

[CP_SWS_Mka_00050] Consideration for ICV calculation at transmission side

Status: DRAFT
Upstream requirements: FO_RS_MACsec_00002, FO_RS_MACsec_00008

⌈If the MKA module needs to calculate an ICV, then the MKA module shall consider the following Ethernet frame parts for the calculation:

- MAC destination address: configured at the corresponding MkaKay as MkaDstMacAddress
- MAC source address: configured at the associated EthCtrlConfig as EthCtrlPhyAddress
- EtherType, specified in [5, IEEE-802.1X-2020] as 0x888E (EAPOL Ethernet type)
- MKPDU without ICV, generated by the MKA module

⌋

[CP_SWS_Mka_CONSTR_00024] Consume MAC source address of a MKPDU via meta data at reception side

Status: DRAFT
Upstream requirements: FO_RS_MACsec_00002, FO_RS_MACsec_00008

⌈At each MkaPaeInstance the MkaPaeRxPdu shall have the meta data item ETHERNET_MAC_64 configured.⌋

[CP_SWS_Mka_00051] Consideration for ICV validation at reception side

Status: DRAFT
Upstream requirements: FO_RS_MACsec_00002, FO_RS_MACsec_00008

⌈If the MKA module needs to validate an ICV, then the MKA module shall consider the following Ethernet frame parts for the validation:

- MAC destination address: configured at the corresponding MkaKay as MkaDstMacAddress identified via the given MkaPaeRxPduId as PduId in context of the Mka_RxIndication call
- MAC source address: given via the meta data item ETHERNET_MAC_64 at the PduInfoPtr in context of the Mka_RxIndication call
- EtherType, specified in [5, IEEE-802.1X-2020] as 0x888E (EAPOL Ethernet type)

- MKPDU without ICV, given with PduInfoPtr in context of the Mka_RxIndication call

⌋

[CP_SWS_Mka_00039] Call of LSduR_MkaTransmit for a generated MKPDU

Status: DRAFT
Upstream requirements: FO_RS_MACsec_00002, FO_RS_MACsec_00008

⌈If the MKA module has generated a MKPDU for a specific MkaPaeInstance and this MKPDU needs to be transmitted, then the MKA module shall call LSduR_MkaTransmit with TxPduId set to MkaPaeTxPduId that is configured for this MkaPaeInstance and SduDataPtr of PduInfoPtr set to the memory address of the generated MKPDU.⌋

Initialization and update of a MACsec Entity (SecY) need to be robust and consider hardware issues. API calls may be repeated multiple times to over come race conditions if hardware of the MACsec Entity react not as expected. The MKA module is responsible to retry function calls once a function to initialize, update or maintain the MACsec Entity returns with E_NOT_OK.

[CP_SWS_Mka_00040] Retry to trigger transmission for a call of LSduR_MkaTransmit

Status: DRAFT
Upstream requirements: FO_RS_MACsec_00002, FO_RS_MACsec_00008

⌈If the MKA module has triggered a transmission of a MKPDU for a specific MkaPaeInstance and call of LSduR_MkaTransmit returns with E_NOT_OK, then the MKA module shall retry to trigger the transmission during the next main cycles again for this MkaPaeInstance until LSduR_MkaTransmit returns with E_OK or Mka_LinkStateChange with TransceiverLinkState set to ETHTRCV_LINK_STATE_DOWN and MkaPaeIdx set to the affected MkaPaeInstance.⌋

[CP_SWS_Mka_00041] Retry of transmission for a generated MKPDU where the corresponding Mka_TxConfirmation returns with result set to E_NOT_OK

Status: DRAFT
Upstream requirements: FO_RS_MACsec_00002, FO_RS_MACsec_00008

⌈If the MKA module triggered a transmission of a MKPDU for a specific MkaPaeInstance where the call of LSduR_MkaTransmit returned with E_OK and the indication of Mka_TxConfirmation with TxPduId set to the MkaPaeTxPduId that is used for this transmission process returns with result set to E_NOT_OK, then the MKA module shall retry the transmission process during the next main cycles again for this MkaPaeInstance until this transmission process is successfully finalized or Mka_LinkStateChange with TransceiverLinkState set to ETHTRCV_LINK_STATE_DOWN and MkaPaeIdx set to the affected MkaPaeInstance.⌋

MKPDUs are transmitted to an IEEE specified PAE address (see [5, IEEE-802.1X-2020] - chapter 11.1.1 (Table 11-1)). The PAE address represents a MAC multicast

address. The MKA module need to update the MAC filter at the affected Ethernet controller to enable the reception of MKPDUs. The update of the MAC address filter should be done as early as possible after the AUTOSAR stack is initialized and the BSW modules are running, e.g. first call of the `Mka_MainFunction`.

[CP_SWS_Mka_00042] Update the MAC address filter in the Ethernet controller with the configured MAC multicast address for MKPDU exchange

Status: DRAFT
Upstream requirements: FO_RS_MACsec_00002, FO_RS_MACsec_00008

⌈After initialization and startup of the MKA module, the MKA module shall update the MAC address filter as early as possible for each different `EthIfController` that is referenced by the configured `MkaPaeInstance`s via `MkaEthIfControllerRef` by calling `EthIf_UpdatePhysAddrFilter` with `CtrlIdx` set to index of the affected `EthIfController`, `PhysAddrPtr` set to the MAC multicast address for MKPDU exchange (see [5, IEEE-802.1X-2020]) configured via `MkaDstMacAddress` of the according `MkaPaeInstance` and `Action` set to `ETH_ADD_TO_FILTER`.⌋

[CP_SWS_Mka_00043] Retry to update the MAC address filter if `EthIf_UpdatePhysAddrFilter` returned with `E_NOT_OK`

Status: DRAFT
Upstream requirements: FO_RS_MACsec_00002, FO_RS_MACsec_00008

⌈If a call of `EthIf_UpdatePhysAddrFilter` to update the MAC address filter returned with `E_NOT_OK` (see [CP_SWS_Mka_00042]), then the MKA module shall retry to update MAC address filter with a call of `EthIf_UpdatePhysAddrFilter` for the affected `EthIfController` latest in the next main function call.⌋

7.7 Communication with MACsec Entity (SecY)

The MKA Module, and particularly a specific MKA Entity (KaY) running over a PAE, shall initialize and maintain a Secure Channel for MACsec over an specific MACsec Entity (SecY). Once a Secure Channel is established for MACsec, multiple Secure Associations could be created and maintained. Each single Secure Associations has its own SAK. The MKA module is responsible to create SAKs and share it with the SecYs participating the same Connectivity Association. The mentioned MACsec Entity can be a SW implementation of MACsec in lower layers or a HW implementation in a Phy, Ethernet Switches or Ethernet Controllers.

The following APIs are used to establish a Secure Channel and to establish Secure Associations. Please refer to [7, SWS_EthIf] Chapter 8 *"API specification"* and to Chapter 8 of this document for more details:

- APIs to establish a Secure Channel, if MACsec processing is performed by an PHY:
  - `EthIf_MacSecInitRxSc`

  - EthIf_MacSecUpdateSecY
  - Mka_MacSecUpdateSecYNotification
- APIs to establish a Secure Channel, if MACsec processing is performed by an Ethernet switch port:
  - EthIf_SwitchMacSecInitRxSc
  - EthIf_SwitchMacSecUpdateSecY
  - Mka_MacSecUpdateSecYNotification
- APIs to establish and maintain a Secure Association, if MACsec processing is performed by an PHY:
  - EthIf_MacSecAddRxSa
  - Mka_MacSecAddRxSaNotification
  - EthIf_MacSecAddTxSa
  - Mka_MacSecAddTxSaNotification
  - EthIf_MacSecSetControlledPortEnabled
  - EthIf_MacSecUpdateTxSa
  - EthIf_MacSecGetTxSaNextPn
  - EthIf_MacSecUpdateRxSa
- APIs to establish and maintain a Secure Association, if MACsec processing is performed by an Ethernet switch port:
  - EthIf_SwitchMacSecAddRxSa
  - Mka_MacSecAddRxSaNotification
  - EthIf_SwitchMacSecAddTxSa
  - Mka_MacSecAddTxSaNotification
  - EthIf_SwitchMacSecSetControlledPortEnabled
  - EthIf_SwitchMacSecUpdateTxSa
  - EthIf_SwitchMacSecGetTxSaNextPn
  - EthIf_SwitchMacSecUpdateRxSa

Initialization and update of a MACsec Entity (SecY) need to be robust and consider hardware issues. API calls may need to be performed multiple times for over come race conditions if hardware of the MACsec Entity react not as expected. The MKA is responsible to retry function calls once a function to initialize, update or maintain the MACsec Entity returns with E_NOT_OK.

[CP_SWS_Mka_00044] Retry of function calls which return with `E_NOT_OK`

Status: DRAFT
Upstream requirements: FO_RS_MACsec_00002, FO_RS_MACsec_00008

⌈If one of the following APIs return with `E_NOT_OK` for a specific `MkaPaeInstance`, then the MKA module shall retry again during the next main cycles again for this `MkaPaeInstance` until `E_OK` is returned or `Mka_LinkStateChange` with `TransceiverLinkState` set to `ETHTRCV_LINK_STATE_DOWN` and `MkaPaeIdx` set to the affected `MkaPaeInstance`:

- `EthIf_MacSecInitRxSc`
- `EthIf_MacSecUpdateSecY`
- `EthIf_SwitchMacSecInitRxSc`
- `EthIf_SwitchMacSecUpdateSecY`
- `EthIf_MacSecAddRxSa`
- `EthIf_MacSecUpdateRxSa`
- `EthIf_MacSecAddTxSa`
- `EthIf_MacSecUpdateTxSa`
- `EthIf_MacSecGetTxSaNextPn`
- `EthIf_MacSecSetControlledPortEnabled`
- `EthIf_MacSecGetMacSecStatistics`
- `EthIf_SwitchMacSecGetMacSecStatistics`
- `EthIf_MacSecOperational`

⌋

[CP_SWS_Mka_00024]

Status: DRAFT
Upstream requirements: FO_RS_MACsec_00002, FO_RS_MACsec_00003

⌈The MKA Module shall propagate the MACsec Entity specific parameters as needed by means of the EthIf API.
This requirement applies for the initialization, shutdown and, maintenance of MACsec related parameters.⌋

[CP_SWS_Mka_00045] Considerations to provide a SAK key reference of type `Mka_SakKeyPtrType`

Status: DRAFT
Upstream requirements: FO_RS_MACsec_00002, FO_RS_MACsec_00008

⌈If the MKA module need to provide a SAK key reference of type `Mka_SakKeyPtrType` for a particular `MkaPaeInstance`, then the MKA module shall consider the following points:

- If MkaCalculateSakHashKey of the referenced MkaPaeConfiguration is set to TRUE, then the MKA module shall calculate a hash key and set the pointer HashKeyPtr to the corresponding memory address. Otherwise set the pointer to NULL_PTR. The MKA module shall calculate this 128 bit value by performing an AES-ECB block encryption operation using SAK as a key over 128 zero bits block.
- If the current cipher suite uses extended packet numbering (XPN), then the MKA module shall calculate a salt key according to [4, IEEE 802.1AE-2018] (see chapter "SAK creation") and set the pointer SaltKeyPtr to the corresponding memory address. Otherwise set the pointer to NULL_PTR.
- The MKA module shall trigger to calculate a SAK and set the pointer SakKeyPtr to the corresponding memory address.

⌋

[CP_SWS_Mka_00025]
Status: DRAFT
Upstream requirements: FO_RS_MACsec_00009

⌈The MKA Module shall collect the MACsec Entity (SecY) statistics when requested by means of the EthIf API.⌋

Note: Other modules may request port specific MACsec statistics in order to set DTCs, answer to Diagnostics, and for monitoring.

[CP_SWS_Mka_00037]
Status: DRAFT
Upstream requirements: FO_RS_MACsec_00001, FO_RS_MACsec_00002

⌈When Mka_GetMacSecStatistics is requested, it shall call the EthIf_MacSecGetMacSecStatistics and it shall wait for the notification of Mka_GetMacSecStatisticsNotification, then it shall trigger the callback configured by MkaGetMacSecStatisticsCallbackNotification.([CP_SWS_Mka_91036])⌋

7.8 Configurable behavior of MKA

[CP_SWS_Mka_00026]
Status: DRAFT
Upstream requirements: FO_RS_MACsec_00005

⌈The status and behavior of a specific MkaPaeInstance shall be configurable per initial configuration.⌋

Note: In case the proposal from [5, IEEE-802.1X-2020] chapter 12.5 is used, the variable *useEAP* is currently not supported (that means, the value shall be per default set to *Never*).

[CP_SWS_Mka_CONSTR_00022] **`MkaPaeInstance` shall either reference an `EthIfController` or an `EthSwtPort`**
Status: DRAFT
Upstream requirements: FO_RS_MACsec_00002

⌈A `MkaPaeInstance` shall either reference an `EthIfController` or an `EthSwtPort`. All other configurations shall be rejected as invalid.⌋

[CP_SWS_Mka_00027]
Status: DRAFT
Upstream requirements: FO_RS_MACsec_00005

⌈It shall be possible to set the configuration of a specific PAE dynamically through the MKA module API.
The change shall apply in the next power cycle.
If a configuration parameter (e.g. through `Mka_SetPaePermissiveMode`, `Mka_SetCknStatus`, or `Mka_SetEnable`) is changed by means of the API, the original per configuration set value shall be ignored and the in NVRAM persisted value shall be used from the next power cycle onwards.⌋

[CP_SWS_Mka_00028]
Status: DRAFT
Upstream requirements: FO_RS_MACsec_00005

⌈In case `MkaOnFailPermissiveMode` == `MKA_PERMISSIVE_MODE_TIMEOUT` and `MkaParticipantActivate` == TRUE for a specific `MkaKayParticipant`, it shall determine that the MKA has failed when all these conditions are met:

- MKA sequence did not succeed → The participants could not reach the state in which the SAK is installed for Rx and Tx.
- MKA timeout (`MkaOnFailPermissiveModeTimeout`) is exceeded.

If all these conditions are met, the error MKA_E_TIMEOUT shall be triggered.⌋

Note: The MKA timeout value is set per configuration with the `MkaOnFailPermissiveModeTimeout` parameter.

[CP_SWS_Mka_00029]
Status: DRAFT
Upstream requirements: FO_RS_MACsec_00005

⌈The MKA timer for `MkaOnFailPermissiveModeTimeout` shall start counting after LinkUp (*ETH_MODE_ACTIVE*) of the referred physical or virtual port.⌋

[CP_SWS_Mka_00030]
Status: DRAFT
Upstream requirements: FO_RS_MACsec_00005

⌈The `MkaOnFailPermissiveModeTimeout` value shall be reset if any of the following conditions is met:

- After start up.
- On the transition from LinkDown (*ETH_MODE_DOWN*) to LinkUp (*ETH_MODE_ACTIVE*) of the referred physical or virtual port.

⌋

7.8.1 MKA behavior

[CP_SWS_Mka_00031] Trigger to configure a MACsec protected port if link state changes to ACTIVE

Status: DRAFT
Upstream requirements: FO_RS_MACsec_00002

⌈If `Mka_LinkStateChange` is called with `TransceiverLinkState` set to `ETHTRCV_LINK_STATE_ACTIVE`, then the MKA module shall trigger the configuration of a MACsec protected port that corresponds to the `MkaPaeInstance` identified with the given `MkaPaeIdx` by immediately starting the MKA sequence(s) via the configured `MkaPaeTxPdu` and `MkaPaeRxPdu` of this `MkaPaeInstance`.⌋

[CP_SWS_Mka_00032] Signal a successful finalization of a configuration for a MACsec protected port

Status: DRAFT
Upstream requirements: FO_RS_MACsec_00002

⌈If the configuration of a MACsec protected port has been successfully finalized, then the MKA module shall immediately perform either one of the following conditions:

- If the `MkaPaeInstance` that corresponds to the MACsec protected port references an `EthIfController` via `MkaEthIfControllerRef`, then call `EthIf_MacSecOperational` with `CtrlIdx` set to this `EthIfController` and with `MacSecOperational` set to `TRUE`.
- If the `MkaPaeInstance` that corresponds to the MACsec protected port references an `EthSwtPort` via `MkaSwitchPortRef`, then call `EthIf_SwitchMacSecOperational` with a `MgmtInfoPtr` that has `SwitchPortIdx` set to this `EthSwtPort` and `SwitchIdx` to the corresponding `EthIfSwitch`, and with `MacSecOperational` set to `TRUE`.

⌋

Note:

- It is sufficient to call `EthIf_MacSecOperational` only once every time after a successful configuration of a MACsec protected port, since the MACsec operational state for the same MACsec protected port is shared across all affected `EthIfController` in the EthIf module.

- A MKA Entity determines that the MACsec protected port is successfully configured as soon as MACsec protected frames can be transmitted and received from both participants.

[CP_SWS_Mka_00047] Shutdown MACsec operational functionality if `MkaPaeInstance` references an `EthIfController`

Status: DRAFT
Upstream requirements: FO_RS_MACsec_00002

⌈When ever the MKA module detects for a specific `MkaPaeInstance` (MACsec protected port) that MACsec protection is violated (e.g. link partner loss) and this `MkaPaeInstance` references an `EthIfController` via `MkaEthIfControllerRef`, then MKA shall shutdown the MACsec operational functionality by considering at least the following points:

- release all resources that belongs to this `MkaPaeInstance`
- disable the MACsec protected port by calling `EthIf_MacSecSetControlledPortEnabled` with `CtrlIdx` set to the referenced `EthIfController` and `ControlledPortEnabled` set to `FALSE`
- remove the receive security channel by calling `EthIf_MacSecResetRxSc` with `CtrlIdx` set to the referenced `EthIfController` and `Sci` set to the id of affected receive security channel
- delete all existing security associations (old and latest) for both directions (reception and transmission) by calling the following APIs:
  - call `EthIf_MacSecUpdateRxSa` and `EthIf_MacSecUpdateTxSa` with `CtrlIdx` set to the referenced `EthIfController` and `Active` set to `FALSE`
  - call `EthIf_MacSecDeleteRxSa` and `EthIf_MacSecDeleteTxSa` with `CtrlIdx` set to the referenced `EthIfController` and `An` set to the affected association number
- call `EthIf_MacSecOperational` with `CtrlIdx` set to the referenced `EthIfController` and `MacSecOperational` set to `FALSE`

⌋

[CP_SWS_Mka_00048] Shutdown MACsec operational functionality if `MkaPaeInstance` references an `EthSwtPort`

Status: DRAFT
Upstream requirements: FO_RS_MACsec_00002

⌈When ever the MKA module detects for a specific `MkaPaeInstance` (MACsec protected port) that MACsec protection is violated (e.g. link partner loss) and this `MkaPaeInstance` references an `EthSwtPort` via `MkaSwitchPortRef`, then MKA shall shutdown the MACsec operational functionality by considering at least the following points:

- release all resources that belongs to this MkaPaeInstance
- disable the MACsec protected port by calling EthIf_SwitchMacSecSet-ControlledPortEnabled with MgmtInfoPtr set to the referenced EthSwt-Port and the corresponding EthSwtPort, and ControlledPortEnabled set to FALSE
- remove the receive security channel by calling EthIf_SwitchMacSecReset-RxSc with with MgmtInfoPtr set to the referenced EthSwtPort and the corresponding EthSwtPort, and Sci set to the id of affected receive security channel
- delete all existing security associations (old and latest) for both directions (reception and transmission) by calling the following APIs:
  - call EthIf_SwitchMacSecUpdateRxSa and EthIf_SwitchMacSec-UpdateTxSa with MgmtInfoPtr set to the referenced EthSwtPort and the corresponding EthSwtPort, and Active set to FALSE
  - call EthIf_SwitchMacSecDeleteRxSa and EthIf_SwitchMacSec-DeleteTxSa with MgmtInfoPtr set to the referenced EthSwtPort and the corresponding EthSwtPort, and An set to the affected association number
- call EthIf_SwitchMacSecOperational with MgmtInfoPtr set to the referenced EthSwtPort and the corresponding EthSwtPort, and MacSec-Operational set to FALSE

⌋

[CP_SWS_Mka_00049] Trigger to reset the configuration of a MACsec protected port if link state changes to DOWN

Status: DRAFT
Upstream requirements: FO_RS_MACsec_00002

⌈If Mka_LinkStateChange is called with TransceiverLinkState set to ETHTRCV_LINK_STATE_DOWN, then the MKA module shall reset the configuration of the MACsec protected port that corresponds to the MkaPaeInstance identified with the given MkaPaeIdx by considering to shutdown the MACsec protected port with respect to the configuration specified in [CP_SWS_Mka_00047] and [CP_SWS_Mka_00048].⌋

7.8.2 MKA Error Handling

To ease complexity of the implementation, the MKA module may heal certain production errors automatically at start-up.

[CP_SWS_Mka_00033]
Status: DRAFT
Upstream requirements: FO_RS_MACsec_00024

⌈If one or more CAKs referenced by `MkaCryptoCknCakKeyRef` is/are not initialized, MKA_E_KEY_NOT_PRESENT_INSTANCE shall be set to "Fail".⌋

Note: Also see CSM Return Code CRYPTO_E_KEY_NOT_AVAILABLE.

[CP_SWS_Mka_00034]
Status: DRAFT
Upstream requirements: FO_RS_MACsec_00020, FO_RS_MACsec_00033

⌈If the verification of the ICV of MKPDUs or the unwrapping of keys fails for one or more CKNs because of a wrong key, MKA_E_KEY_MISMATCH_INSTANCE shall be set to "Fail".⌋

[CP_SWS_Mka_00035]
Status: DRAFT
Upstream requirements: FO_RS_MACsec_00032

⌈If the MKA peer only supports incompatible cipher suites, MKA_E_ALGO_MISMATCH_INSTANCE shall be set to "Fail".⌋

[CP_SWS_Mka_00036]
Status: DRAFT
Upstream requirements: FO_RS_MACsec_00005, FO_RS_MACsec_00006

⌈MKA_E_TIMEOUT_INSTANCE, MKA_E_KEY_NOT_PRESENT_INSTANCE, MKA_E_KEY_MISMATCH_INSTANCE, and MKA_E_ALGO_MISMATCH_INSTANCE shall be healed (set to "Pass"), when the error condition is not applicable anymore.⌋

Note: If an implementer chooses to not implement [CP_SWS_Mka_00036], the mentioned errors shall be healed on start-up of the MKA module.

7.9 Error Classification

Chapter [2, General Specification of Basic Software Modules] 7.2 *"Error Handling"* describes the error handling of the Basic Software in detail. Above all, it constitutes a classification scheme consisting of the following error types which may occur in BSW modules.

Based on this foundation, the following section specifies particular errors arranged in the respective subsections below.

7.9.1 Development Errors

[CP_SWS_Mka_91035] Definition of development errors in module Mka

Status: DRAFT
Upstream requirements: SRS_BSW_00337, SRS_BSW_00385, SRS_BSW_00323

⌈

Type of error	*Related error code*	*Error value*
MKA Component initiated with null configuration	MKA_E_CFG_NULL_PTR	0x01
API called with invalid parameter value.	MKA_E_INVALID_PARAMETER	0x04
API called with a NULL pointer.	MKA_E_PARAM_POINTER	0x05
API called prior module is initialized.	MKA_E_UNINIT	0x06

⌋

7.9.2 Runtime Errors

There are no runtime errors.

7.9.3 Production Errors

7.9.3.1 MKA_E_TIMEOUT_INSTANCE

[CP_SWS_Mka_00200] MKA timeout while negotiating with remote peer.

Upstream requirements: SRS_BSW_00385, SRS_BSW_00337

⌈

Diagnostic Event (Error Name)	MKA_E_TIMEOUT_INSTANCE
Description	In case MkaOnFailPermissiveMode == MKA_PERMISSIVE_MODE_TIMEOUT and MkaOnFailPermissiveModeTimeout is overflowed, the error will be set.
Failed condition	If the PAE instance's MkaOnFailPermissiveMode == TIMEOUT and MkaOnFail PermissiveModeTimeout is overflowed, the error will be set.
Passed condition	If the PAE instance's MkaOnFailPermissiveMode == NEVER or If the PAE instance's MkaOnFailPermissiveMode == TIMEOUT, and the MkaKay could establish a transmission and reception SC with a peer before MkaOnFailPermissiveModeTimeout is reached, the error is not set.

⌋

7.9.3.2 MKA_E_KEY_NOT_PRESENT_INSTANCE

[CP_SWS_Mka_00201] Necessary keys not present to initiate MKA negotiation.

Upstream requirements: SRS_BSW_00385, SRS_BSW_00337

⌈

Diagnostic Event (Error Name)	MKA_E_KEY_NOT_PRESENT_INSTANCE
Description	Pre-shared keys (i.e. CAK, ICK or KEK) to start the MKA sequence are not present in at least one of the active configured Kay Participants.
Failed condition	The pre-shared keys of an active MkaKayParticipant are not present.
Passed condition	All Kay participants have the needed pre-shared keys present.

⌋

7.9.3.3 MKA_E_KEY_MISMATCH_INSTANCE

[CP_SWS_Mka_00202] MKA negotiation failed due to key mismatch with remote peer (MKPDUs ICV mismatch).

Upstream requirements: SRS_BSW_00385, SRS_BSW_00337

⌈

Diagnostic Event (Error Name)	MKA_E_KEY_MISMATCH_INSTANCE
Description	Triggered in case MKPDU cannot be validated for received MKPDUs which distribute a matching CKN.
Failed condition	A received MKPDU with matching CKN cannot be successfully validated.
Passed condition	All received MKPDUs with matching CKN are successfully validated.

⌋

7.9.3.4 MKA_E_ALGO_MISMATCH_INSTANCE

[CP_SWS_Mka_00203] MKA negotiation failed due to incompatible cipher suite with remote peer.

Upstream requirements: SRS_BSW_00385, SRS_BSW_00337

⌈

Diagnostic Event (Error Name)	MKA_E_ALGO_MISMATCH_INSTANCE
Description	Triggered in case the participants in the MKA communication do not support any MACsec cipher suite in common and therefore cannot distribute neither install a valid SAK.
Failed condition	The KaY participants of a communication (local and remote) do not support a common MACsec cipher suite.
Passed condition	The KaY participants of a communication (local and remote) support one or more common MACsec cipher suites.

⌋

7.10 Security Events

[CP_SWS_Mka_00301] Security event report

Upstream requirements: FO_RS_MACsec_00009

⌈If security event reporting has been enabled for the MKA module (MkaEnableSecurityEventReporting = true) the security events shall be reported to the IdsM via the interfaces defined in [2, CP-SWS-BSWGeneral]⌋

[CP_SWS_Mka_00302] Security events for Mka ⌈

Name	Description	ID
SEV_MKA_AUTHENTICATION_FAILURE	Event triggered when the authentication during the MKA communication has failed (wrong CKN/CAK).	78
SEV_MKA_TIMEOUT	Event triggered when the timeout for the MKA communication has expired.	79
SEV_MKA_PORT_NOT_ENABLED	Event triggered when the indicated port for the MKA communication is not enable.	80
SEV_MKA_CIPHER_SUITE_NOT_SUPPORTED	Event triggered when there is no Cipher Suite supported.	81
SEV_MKA_PORT_NUMBER_CHANGE	Event triggered when during the MKA communication the port number has changed.	82

⌋

[CP_SWS_Mka_00309] Security event context data definition: SEV_MKA_AUTHENTICATION_FAILURE ⌈

SEV Name	SEV_MKA_AUTHENTICATION_FAILURE	
ID	78	
Description	Event triggered when the authentication during the MKA communication has failed (wrong CKN/CAK).	
Context Data Version	1	
Context Data	Data Type	Allowed Values
PortId	uint8 [2]	
CKN	uint8 [32]	
MACAdressOfPeer	uint8 [6]	

⌋

[CP_SWS_Mka_00310] Security event context data definition: SEV_MKA_TIMEOUT ⌈

SEV Name	SEV_MKA_TIMEOUT	
ID	79	
Description	Event triggered when the timeout for the MKA communication has expired.	
Context Data Version	1	
Context Data	Data Type	Allowed Values
PortId	uint8 [2]	
CKN	uint8 [32]	
MACAdressOfPeer	uint8 [6]	

⌋

[CP_SWS_Mka_00311] Security event context data definition: SEV_MKA_PORT_NOT_ENABLED ⌈

SEV Name	*SEV_MKA_PORT_NOT_ENABLED*	
ID	80	
Description	Event triggered when the indicated port for the MKA communication is not enable.	
Context Data Version	1	
Context Data	*Data Type*	*Allowed Values*
PortId	uint8 [2]	
CKN	uint8 [32]	
MACAdressOfPeer	uint8 [6]	

⌋

[CP_SWS_Mka_00312] Security event context data definition: SEV_MKA_CIPHER_SUITE_NOT_SUPPORTED ⌈

SEV Name	*SEV_MKA_CIPHER_SUITE_NOT_SUPPORTED*	
ID	81	
Description	Event triggered when there is no Cipher Suite supported.	
Context Data Version	1	
Context Data	*Data Type*	*Allowed Values*
PortId	uint8 [2]	
CKN	uint8 [32]	
MACAdressOfPeer	uint8 [6]	

⌋

[CP_SWS_Mka_00313] Security event context data definition: SEV_MKA_PORT_NUMBER_CHANGE ⌈

SEV Name	*SEV_MKA_PORT_NUMBER_CHANGE*	
ID	82	
Description	Event triggered when during the MKA communication the port number has changed.	
Context Data Version	1	
Context Data	*Data Type*	*Allowed Values*
PortId	uint8 [2]	
CKN	uint8 [32]	
MACAdressOfPeer	uint8 [6]	

⌋

[CP_SWS_Mka_00304] Description of security event SEV_MKA_AUTHENTICATION_FAILURE

Upstream requirements: FO_RS_MACsec_00009

⌈MKA shall raise the SEv `SEV_MKA_AUTHENTICATION_FAILURE` to the IdsM, when MKA module dropped a packet, because of authentication failure.⌋

[CP_SWS_Mka_00305] Description of security event SEV_MKA_TIMEOUT

Upstream requirements: FO_RS_MACsec_00009

⌈MKA module shall raise the SEv `SEV_MKA_TIMEOUT` to the IdsM, when `MkaOnFail-PermissiveModeTimeout` has expired without receiving an MKA packet.⌋

[CP_SWS_Mka_00306] Description of security event SEV_MKA_PORT_NOT_ENABLED

Upstream requirements: FO_RS_MACsec_00009

⌈MKA module shall raise the SEv `SEV_MKA_PORT_NOT_ENABLED` to the IdsM, when MKA module cannot communicate with the assigned port anymore.⌋

[CP_SWS_Mka_00307] Description of security event SEV_MKA_CIPHER_SUITE_NOT_SUPPORTED

Upstream requirements: FO_RS_MACsec_00009

⌈MKA module shall raise the SEv `SEV_MKA_CIPHER_SUITE_NOT_SUPPORTED` to the IdsM, when during the MKA communication, the MKA module detects a usage of a non supported Cipher Suite.⌋

[CP_SWS_Mka_00308] Description of security event SEV_MKA_PORT_NUMBER_CHANGE

Upstream requirements: FO_RS_MACsec_00009

⌈MKA module shall raise the SEv `SEV_MKA_PORT_NUMBER_CHANGE` to the IdsM, when during an established MKA communication, the MKA module detects that the used port from the peer has changed.⌋

8 API specification

8.1 Imported types

In this chapter all types included from the following files are listed.

[CP_SWS_Mka_91005] Definition of imported datatypes of module Mka

Status: DRAFT

⌈

Module	*Header File*	*Imported Type*
Can	Can_GeneralTypes.h	Can_ControllerStateType
Comtype	ComStack_Types.h	PduIdType
	ComStack_Types.h	PduInfoType
	ComStack_Types.h	PduLengthType
Csm	Rte_Csm_Type.h	Crypto_OperationModeType
	Rte_Csm_Type.h	Crypto_VerifyResultType
Dem	Rte_Dem_Type.h	Dem_EventIdType
	Rte_Dem_Type.h	Dem_EventStatusType
Eth	Eth_GeneralTypes.h	Eth_FilterActionType
EthSwt	Eth_GeneralTypes.h	EthSwt_MgmtInfoType
EthTrcv	Eth_GeneralTypes.h	EthTrcv_LinkStateType
IdsM	IdsM.h	IdsM_SecurityEventIdType
	Rte_IdsM_Type.h	IdsM_TimestampDataType
NvM	Rte_NvM_Type.h	NvM_BlockIdType
Std	Std_Types.h	Std_ReturnType

⌋

8.2 Type definitions

8.2.1 Mka_MacSecConfigType

[CP_SWS_Mka_91002] Definition of datatype Mka_MacSecConfigType

Status: DRAFT

⌈

Name	Mka_MacSecConfigType (draft)	
Kind	Structure	
Elements	ProtectFrames	
	Type	boolean
	Comment	Indicates status if the MACsec protection of the frames is active or not
	ReplayProtect	
	Type	boolean

▽

△

	Comment	Indicates status if replay protection is enable or disable
	ReplayWindow	
	Type	uint32
	Comment	If ReplayProtect is enable, indicates the used replay protect window
	ValidateFrames	
	Type	Mka_ValidateFramesType
	Comment	Status of the validation of the frames. See Mka_ValidateFramesType for possible values
	CurrentCipherSuite	
	Type	uint64
	Comment	Indicates which cipher suite is used in the SecY to update. The Cipher Suite Identifiers are defined in Table 14-1 of [4, IEEE-802.1AE-2018].
	ConfidentialityOffset	
	Type	Mka_ConfidentialityOffsetType
	Comment	Set the Confidentiality Offset. See Mka_ConfidentialityOffsetType for possible values
	ControlledPortEnabled	
	Type	boolean
	Comment	Status if the controlled port is enabled or disabled
Description	Structure to configure a referred SecY **Tags:** atp.Status=draft	
Available via	Mka.h	

⌋

8.2.2 Mka_ValidateFramesType

[CP_SWS_Mka_91004] Definition of datatype Mka_ValidateFramesType

Status: DRAFT

⌈

Name	Mka_ValidateFramesType (draft)		
Kind	Enumeration		
Range	MKA_VALIDATE_ DISABLED	0	Disable validation, remove SecTAGs and ICVs (if present) from received frames.
	MKA_VALIDATE_ CHECKED	1	Enable validation, do not discard invalid frames
	MKA_VALIDATE_STRICT	2	Enable validation and discard invalid frames
Description	Controls validation of received frames **Tags:** atp.Status=draft		
Available via	Mka.h		

⌋

8.2.3 Mka_ConfidentialityOffsetType

[CP_SWS_Mka_91003] Definition of datatype Mka_ConfidentialityOffsetType

Status: DRAFT

⌈

Name	Mka_ConfidentialityOffsetType (draft)		
Kind	Enumeration		
Range	MKA_CONFIDENTIALITY_NONE	0	Confidentiality protection disabled
	MKA_CONFIDENTIALITY_OFFSET_0	1	Zero initial octets of each user data without confidentiality protection
	MKA_CONFIDENTIALITY_OFFSET_30	2	30 initial octets of each user data without confidentiality protection
	MKA_CONFIDENTIALITY_OFFSET_50	3	50 initial octets of each user data without confidentiality protection
Description	Indicates the offset in case of integrity with confidentiality **Tags:** atp.Status=draft		
Available via	Mka.h		

⌋

8.2.4 Mka_Stats_Tx_SecYType

[CP_SWS_Mka_91008] Definition of datatype Mka_Stats_Tx_SecYType

Status: DRAFT

⌈

Name	Mka_Stats_Tx_SecYType (draft)	
Kind	Structure	
Elements	OutPkts_Untagged	
	Type	uint64
	Comment	The number of packets transmitted without a SecTAG because secy ProtectFramesEnable is configured false
	OutPkts_TooLong	
	Type	uint64
	Comment	The number of transmitted packets discarded because their length is greater than the accepted maximum length (mtu) of the Port
	OutOctets_Protected	
	Type	uint64
	Comment	The number of plain text octets integrity protected but not encrypted in transmitted frames
	OutOctets_Encrypted	
	Type	uint64
	Comment	The number of plain text octets integrity protected and encrypted in transmitted frames

▽

△

Description	MACsec Entity (SecY) transmission statistics **Tags:** atp.Status=draft
Available via	Mka.h

⌋

8.2.5 Mka_Stats_Rx_SecYType

[CP_SWS_Mka_91010] Definition of datatype Mka_Stats_Rx_SecYType

Status: DRAFT

⌈

Name	Mka_Stats_Rx_SecYType (draft)	
Kind	Structure	
Elements	InPkts_Untagged	
	Type	uint64
	Comment	The number of packets without the MACsec tag (SecTAG) received if Mka_ValidateFrames was not 'MKA_VALIDATE_STRICT'
	InPkts_NoTag	
	Type	uint64
	Comment	The number of received packets without a SecTAG discarded because Mka_ValidateFrames was 'MKA_VALIDATE_STRICT'
	InPkts_BadTag	
	Type	uint64
	Comment	The number of received packets discarded with an invalid SecTAG, zero value PN, or invalid ICV
	InPkts_NoSa	
	Type	uint64
	Comment	The number of received packets with an unknown SCI or for an unused SA by the security entity
	InPkts_NoSaError	
	Type	uint64
	Comment	The number of packets discarded because the received SCI is unknown or the SA is not in use
	InPkts_Overrun	
	Type	uint64
	Comment	The number of packets discarded because they exceeded cryptographic performance capabilities
	InOctets_Validated	
	Type	uint64
	Comment	The number of plaintext octets recovered from packets that were integrity protected but not encrypted
	InOctets_Decrypted	
	Type	uint64

▽

△

	Comment	The number of plaintext octets recovered from packets that were integrity protected and encrypted
Description	MACsec Entity (SecY) reception statistics **Tags:** atp.Status=draft	
Available via	Mka.h	

⌋

8.2.6 Mka_Stats_Tx_ScType

[CP_SWS_Mka_91009] Definition of datatype Mka_Stats_Tx_ScType

Status: DRAFT

⌈

Name	Mka_Stats_Tx_ScType (draft)	
Kind	Structure	
Elements	OutPkts_Protected	
	Type	uint64
	Comment	The number of integrity protected but not encrypted packets for this transmit SC
	OutPkts_Encrypted	
	Type	uint64
	Comment	The number of integrity protected and encrypted packets for this transmit SC
Description	Secure Channel transmission statistics **Tags:** atp.Status=draft	
Available via	Mka.h	

⌋

8.2.7 Mka_Stats_Rx_ScType

[CP_SWS_Mka_91011] Definition of datatype Mka_Stats_Rx_ScType

Status: DRAFT

⌈

Name	Mka_Stats_Rx_ScType (draft)	
Kind	Structure	
Elements	InPkts_Ok	
	Type	uint64
	Comment	The number of packets received for this secure channel successfully validated and within the replay window
	InPkts_Unchecked	

▽

△

	Type	uint64
	Comment	The number of packets received for this secure channel, if Mka_ ValidateFrames was 'MKA_VALIDATE_DISABLED'
	InPkts_Delayed	
	Type	uint64
	Comment	The number of received packets, for this secure channel, with packet number (PN) lower than the lowest acceptable packet number (Lowest Pn) and ReplayProtect was false
	InPkts_Late	
	Type	uint64
	Comment	The number of packets discarded, for this secure channel, because the received packet number (PN) was lower than the lowest acceptable packet number (LowestPn) and ReplayProtect was true
	InPkts_Invalid	
	Type	uint64
	Comment	The number of packets, for this secure channel, that failed validation but could be received because ValidateFrames was 'MKA_VALIDATE_ CHECKED' and the data was not encrypted (so the original frame could be recovered)
	InPkts_NotValid	
	Type	uint64
	Comment	The number of packets discarded, for this secure channel, because validation failed and ValidateFrames was 'MKA_VALIDATE_STRICT' or the data was encrypted (so the original frame could not be recovered)
Description	Secure Channel reception statistics **Tags:** atp.Status=draft	
Available via	Mka.h	

⌋

8.2.8 Mka_SakKeyPtrType

[CP_SWS_Mka_91013] Definition of datatype Mka_SakKeyPtrType

Status: DRAFT

⌈

Name	Mka_SakKeyPtrType (draft)	
Kind	Structure	
Elements	HashKeyPtr	
	Type	const uint8*
	Comment	Pointer to the Hash Key. This is a 128 bit value calculated by performing AES-ECB block encryption operation using SAK as a key over 128 zero bits block. This Hash is required for the AES-GCM algorithm calculation as described in NIST SP 800-38D.
	SakKeyPtr	
	Type	const uint8*
	Comment	Pointer to the SAK

▽

△

	SaltKeyPtr	
	Type	const uint8*
	Comment	Pointer to the Salt when the current cipher suite uses extended packet numbering (XPN). Salt calculation is described in IEEE 802.1AE, chapter "SAK creation".
Description	SAK key reference **Tags:** atp.Status=draft	
Available via	Mka.h	

⌋

8.2.9 Mka_PermissiveModeType

[CP_SWS_Mka_91012] Definition of datatype Mka_PermissiveModeType

Status: DRAFT

⌈

Name	Mka_PermissiveModeType (draft)		
Kind	Enumeration		
Range	MKA_PERMISSIVE_MODE_NEVER	0	The controlled port will never be set to enabled if the participants cannot establish and successfully use a MACsec Secure Channel.
	MKA_PERMISSIVE_MODE_TIMEOUT	1	The controlled port will be set to enabled and MACsec will not be used in the referred port if the timeout value (MkaOnFailPermissive Mode Timeout) is reached and none MKA instance under the PAE instance could success the following conditions: - A participant belonging to the same CA was recognized and authenticated. - A secure channel could be established. - Both participants can transmit and receive MACsec protected traffic through the SC.
Description	Permissive modes of MKA **Tags:** atp.Status=draft		
Available via	Mka.h		

⌋

8.2.10 Mka_Stats_SecYType

[CP_SWS_Mka_91028] Definition of datatype Mka_Stats_SecYType

Status: DRAFT

⌈

Name	Mka_Stats_SecYType (draft)	
Kind	Structure	
Elements	StatsTxPhy	
	Type	Mka_Stats_Tx_SecYType
	Comment	Set of statistics in the Security Entity Phy by transmision
	StatsRxPhy	
	Type	Mka_Stats_Rx_SecYType
	Comment	Set of statistics in the Security Entity Phy by reception
	StatsTxSc	
	Type	Mka_Stats_Tx_ScType
	Comment	Set of statistics in the Security Entity's Secure Channel by reception
	StatsRxSc	
	Type	Mka_Stats_Rx_ScType
	Comment	Set of statistics in the Security Entity's Secure Channel by reception
Description	Security Entity statistics **Tags:** atp.Status=draft	
Available via	Mka.h	

⌋

8.2.11 Mka_PaeStatusType

[CP_SWS_Mka_91027] Definition of datatype Mka_PaeStatusType

Status: DRAFT

⌈

Name	Mka_PaeStatusType (draft)	
Kind	Structure	
Elements	ConnectionStatus	
	Type	Mka_MkaStatus
	Comment	Status of the MKA
	PeerSci	
	Type	uint64
	Comment	SCI includes the peer's MAC and port
	CknInUse	
	Type	Array of unsigned char[32]
	Size	32
	Comment	CKN used for the establishment of the MACsec Secure Channel

▽

△

Description	Structure with the specific information of a PAE **Tags:** atp.Status=draft
Available via	Mka.h

⌋

8.2.12 Mka_MkaStatusType

[CP_SWS_Mka_91025] Definition of datatype Mka_MkaStatus

Status: DRAFT

⌈

Name	Mka_MkaStatus (draft)		
Kind	Enumeration		
Range	MKA_STATUS_MACSEC_RUNNING	0	MKA session key has been agreed and MACsec link is currently up
	MKA_STATUS_WAITING_PEER_LINK	1	MKA is waiting for a link up of the underlying device to begin negotiation
	MKA_STATUS_WAITING_PEER	2	MKA is waiting for a remote peer to transmit MKPDU's to begin negotiation
	MKA_STATUS_IN_PROGRESS	3	MKA negotiation is ongoing
	MKA_STATUS_AUTH_FAIL_UNKNOWN_PEER	4	MKA negotiation is not possible because ICV's of remote peer are invalid (ICK and therefore CAK keys are different)
	MKA_STATUS_UNDEFINED	0xFF	Undefined state, reported when the given bus is disabled
Description	Status of the MKA instance. **Tags:** atp.Status=draft		
Available via	Mka.h		

⌋

8.2.13 Mka_ConfigType

[CP_SWS_Mka_91026] Definition of datatype Mka_ConfigType

Status: DRAFT

⌈

Name	Mka_ConfigType (draft)
Kind	Structure
Description	Implementation specific structure of the post build configuration **Tags:** atp.Status=draft
Available via	Mka.h

⌋

8.2.14 Mka_CanSecConfigType

[SWS_Mka_91051] Definition of datatype Mka_CanSecConfigType ⌈

Name	Mka_CanSecConfigType	
Kind	Structure	
Elements	ProtectFrames	
	Type	boolean
	Comment	Indicates status if the CANsec protection of the frames is active or not
	ReplayProtect	
	Type	boolean
	Comment	Indicates status if replay protection is enabled or disabled
	ReplayWindow	
	Type	uint32
	Comment	If ReplayProtect is enabled, indicates the used replay protect window
	ValidateFrames	
	Type	Mka_ValidateFramesType
	Comment	Status of the validation of the frames. See Mka_ValidateFramesType for possible values
	CurrentCipherSuite	
	Type	uint64
	Comment	Indicates which cipher suite is used in the SecY to update. The Cipher Suite Identifiers are defined in Table 14-1 of [4, IEEE-802.1AE-2018].
	ConfidentialityOffset	
	Type	Mka_ConfidentialityOffsetType
	Comment	Set the Confidentiality Offset. See Mka_ConfidentialityOffsetType for possible values. Hint: MKA_CONFIDENTIALITY_OFFSET_30 and MKA_CONFIDENTIALITY_OFFSET_50 are not supported by [CiA 612-3]. If these values are used, the CANsec SecY shall respond with E_NOT_OK.
	ControlledPortEnabled	
	Type	boolean
	Comment	Status if the controlled port is enabled or disabled
Description	Structure to configure a referred CANsec SecY	
Available via	Mka.h	

⌋

8.3 Function definitions

This chapter contains all APIs provided by the module.

8.3.1 Mka_Init

[CP_SWS_Mka_91001] Definition of API function Mka_Init

Status: DRAFT

⌈

Service Name	Mka_Init (draft)	

Syntax	`void Mka_Init (`
	`  const Mka_ConfigType* ConfigPtr`
	`)`	

Service ID [hex]	0x01	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	ConfigPtr	Points to the implementation specific structure
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	Initializes the MKA module **Tags:** atp.Status=draft	
Available via	Mka.h	

⌋

8.3.2 Mka_SetCknStatus

[CP_SWS_Mka_91015] Definition of API function Mka_SetCknStatus

Status: DRAFT

⌈

Service Name	Mka_SetCknStatus (draft)	

Syntax	`Std_ReturnType Mka_SetCknStatus (`
	`  uint8 MkaPaeIdx,`
	`  boolean Enable,`
	`  const uint8* Ckn,`
	`  uint8 CknLength`
	`)`	

Service ID [hex]	0x03	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different MkaPaeIdx, Non reentrant for the same MkaPaeIdx	
Parameters (in)	MkaPaeIdx	Index of the PAE within the context of the MKA module
	Enable	Boolean to control the Mka Participant Activate status. True -> The MKA Participant exchanges MKPDUs. False -> The MKA Participant does not exchange MKPDUs.
	Ckn	Connectivity Association Key Name to identify the KaY participant
	CknLength	Length of the CKN parameter provided
Parameters (inout)	None	
Parameters (out)	None	

▽

△

Return value	Std_ReturnType	E_OK: The request has been accepted and the Status of the CKN from a PAE has been set. E_NOT_OK: The status could not be set because call to lower layer failed.
Description	Set status of a CKN from a PAE **Tags:** atp.Status=draft	
Available via	Mka.h	

⌋

[CP_SWS_Mka_01001]

Status: DRAFT

⌈The function Mka_SetCknStatus shall set the activation status of the MkaKayParticipant which contains the provided Ckn under the provided MkaPaeIdx.
The new activation status shall be persistently stored in NVM and used from next power cycle onwards (as required in [CP_SWS_Mka_00030]).
The per configuration provided activation status (MkaParticipantActivate) of the MkaKayParticipant shall not be used if a valid value is stored on the NVM.⌋

8.3.3 Mka_GetCknStatus

[CP_SWS_Mka_91016] Definition of API function Mka_GetCknStatus

Status: DRAFT

⌈

Service Name	Mka_GetCknStatus (draft)	
Syntax	`Std_ReturnType Mka_GetCknStatus ( uint8 MkaPaeIdx, const uint8* Ckn, uint8 CknLength, boolean* EnablePtr )`	
Service ID [hex]	0x04	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different MkaPaeIdx, Non reentrant for the same MkaPaeIdx	
Parameters (in)	MkaPaeIdx	Index of the PAE within the context of the MKA module
	Ckn	Connectivity Association Key Name to identify the KaY participant
	CknLength	Length of the CKN parameter provided
Parameters (inout)	None	
Parameters (out)	EnablePtr	Pointer to the Mka Participant activation status
Return value	Std_ReturnType	E_OK: The request has been accepted and EnablePtr is returned (see Parameters (out)) E_NOT_OK: The status could not be retrieved because call to lower layer failed
Description	Get Status of a CKN from a PAE **Tags:** atp.Status=draft	
Available via	Mka.h	

⌋

8.3.4 Mka_SetEnable

[CP_SWS_Mka_91020] Definition of API function Mka_SetEnable

Status: DRAFT

⌈

Service Name	Mka_SetEnable (draft)	
Syntax	`Std_ReturnType Mka_SetEnable ( uint8 MkaPaeIdx, boolean Enable )`	
Service ID [hex]	0x08	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different MkaPaeIdx, Non reentrant for the same MkaPaeIdx	
Parameters (in)	MkaPaeIdx	Index of the PAE within the context of the MKA module
	Enable	Boolean to control the Mka activation of a PAE.
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: The request has been accepted and MKA activation status has been set `E_NOT_OK`: MKA activation status could not be set because call to lower layer failed
Description	Set the MKA activation status of a PAE **Tags:** atp.Status=draft	
Available via	Mka.h	

⌋

[CP_SWS_Mka_01002]

Status: DRAFT

⌈The function `Mka_SetEnable` shall set the activation status of the `MkaKay` of the provided `MkaPaeIdx`.
The new activation status shall be persistently stored in NVM and used from next power cycle onwards (as required in [CP_SWS_Mka_00030]).⌋

8.3.5 Mka_GetEnable

[CP_SWS_Mka_91017] Definition of API function Mka_GetEnable

Status: DRAFT

⌈

Service Name	Mka_GetEnable (draft)
Syntax	`Std_ReturnType Mka_GetEnable ( uint8 MkaPaeIdx, boolean* EnablePtr )`
Service ID [hex]	0x05
Sync/Async	Synchronous

▽

△

Reentrancy	Reentrant for different MkaPaeIdx, Non reentrant for the same MkaPaeIdx	
Parameters (in)	MkaPaeIdx	Index of the PAE within the context of the MKA module
Parameters (inout)	None	
Parameters (out)	EnablePtr	Pointer to the Mka Participant activation status of a PAE
Return value	Std_ReturnType	E_OK: The request has been accepted and EnablePtr is returned (see Parameters (out)) E_NOT_OK: MKA activation status could not be retrieved because call to lower layer failed
Description	Get the MKA activation status of a PAE **Tags:** atp.Status=draft	
Available via	Mka.h	

⌋

8.3.6 Mka_GetPaeStatus

[CP_SWS_Mka_91018] Definition of API function Mka_GetPaeStatus

Status: DRAFT

⌈

Service Name	Mka_GetPaeStatus (draft)	
Syntax	`Std_ReturnType Mka_GetPaeStatus ( uint8 MkaPaeIdx, Mka_PaeStatusType* PaeStatusPtr )`	
Service ID [hex]	0x06	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different MkaPaeIdx, Non reentrant for the same MkaPaeIdx	
Parameters (in)	MkaPaeIdx	Index of the PAE within the context of the MKA module
Parameters (inout)	None	
Parameters (out)	PaeStatusPtr	Pointer to the status structure, which includes detailed information of a PAE.
Return value	Std_ReturnType	E_OK: The request has been accepted and PaeStatusPtr is returned (see Parameters (out)) E_NOT_OK: The status could not be retrieved because call to lower layer failed
Description	Get detailed information of a PAE **Tags:** atp.Status=draft	
Available via	Mka.h	

⌋

8.3.7 Mka_SetPaePermissiveMode

[CP_SWS_Mka_91021] Definition of API function Mka_SetPaePermissiveMode

Status: DRAFT

⌈

Service Name	Mka_SetPaePermissiveMode (draft)	
Syntax	`Std_ReturnType Mka_SetPaePermissiveMode ( uint8 MkaPaeIdx, Mka_PermissiveModeType PermissiveMode )`	
Service ID [hex]	0x09	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different MkaPaeIdx, Non reentrant for the same MkaPaeIdx	
Parameters (in)	MkaPaeIdx	Index of the PAE within the context of the MKA module
	PermissiveMode	Permissive mode to set in the PAE.
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: The request has been accepted and Permissive mode of KaY has been set `E_NOT_OK`: The Permissive mode of KaY could not be set because call to lower layer failed
Description	Set Permissive Mode of a KaY **Tags:** atp.Status=draft	
Available via	Mka.h	

⌋

[CP_SWS_Mka_01003]

Status: DRAFT

⌈The function `Mka_SetPaePermissiveMode` shall set the `PermissiveMode` of the `MkaPaeInstance` referred with the `MkaPaeIdx`.
The new `PermissiveMode` shall be persistently stored in NVM and used from next power cycle onwards (as required in [CP_SWS_Mka_00030]).
The per configuration provided `MkaOnFailPermissiveMode` of the `MkaPaeInstance` shall not be used if a valid value is stored on the NVM.⌋

8.3.8 Mka_StartPae

[CP_SWS_Mka_91022] Definition of API function Mka_StartPae

Status: DRAFT

⌈

Service Name	Mka_StartPae (draft)
Syntax	`Std_ReturnType Mka_StartPae ( uint8 MkaPaeIdx )`

▽

△

Service ID [hex]	0x10	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different MkaPaeIdx, Non reentrant for the same MkaPaeIdx	
Parameters (in)	MkaPaeIdx	Index of the PAE within the context of the MKA module
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: The request has been accepted and PAE instance has been started (In case MkaPaeConfiguration.Autostart = False this method, the PAE operation has been started) `E_NOT_OK`: PAE instance could not be started because call to lower layer failed
Description	Manual start of the PAE instace. (In case MkaPaeConfiguration.Autostart = False this method starts the PAE operation) **Tags:** atp.Status=draft	
Available via	Mka.h	

⌋

[CP_SWS_Mka_01004]

Status: DRAFT

⌈The function `Mka_StartPae` shall start the operation of the `MkaPaeInstance` referred with the `MkaPaeIdx` if the `MkaAutoStart` configuration parameter is TRUE. If the `MkaAutoStart` configuration parameter is FALSE, `Mka_StartPae` will not have any effect on the referred `MkaPaeInstance`.⌋

8.3.9 Mka_GetMacSecStatistics

[CP_SWS_Mka_91019] Definition of API function Mka_GetMacSecStatistics

Status: DRAFT

⌈

Service Name	Mka_GetMacSecStatistics (draft)	
Syntax	`Std_ReturnType Mka_GetMacSecStatistics ( uint8 MkaPaeIdx, Mka_Stats_SecYType* MacSecStatsPtr )`	
Service ID [hex]	0x07	
Sync/Async	Asynchronous	
Reentrancy	Reentrant for different MkaPaeIdx, Non reentrant for the same MkaPaeIdx	
Parameters (in)	MkaPaeIdx	Index of the PAE within the context of the MKA module
Parameters (inout)	None	
Parameters (out)	MacSecStatsPtr	Pointer to a structure including the MACsec statistics of an MKA participant
Return value	Std_ReturnType	`E_OK`: The request has been accepted and MacSec Statistics has been returned. `E_NOT_OK`: MacSec Statistics could not be retrieved because call to lower layer failed

▽

△

Description	Get Statistics of a PAE **Tags:** atp.Status=draft
Available via	Mka.h

⌋

8.3.10 Mka_LinkStateChange

[CP_SWS_Mka_91023] Definition of API function Mka_LinkStateChange

Status: DRAFT

⌈

Service Name	Mka_LinkStateChange (draft)	

Syntax	`Std_ReturnType Mka_LinkStateChange (`
	`  uint8 MkaPaeIdx,`
	`  EthTrcv_LinkStateType TransceiverLinkState`
	`)`	

Service ID [hex]	0x1d	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different MkaPaeIdx, Non reentrant for the same MkaPaeIdx	
Parameters (in)	MkaPaeIdx	Index of the PAE within the context of the MKA module
	TransceiverLinkState	The Ethernet link state of a physical Ethernet connection.
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: The request has been accepted and MKA module has been informed that a dedicated Trcv/Switch proteced port link state has changed and can start with the MKA sequence. `E_NOT_OK`: MKA didn't start the MKA sequence.
Description	To inform MKA that a dedicated Trcv/Switch/PAC port link state has changed **Tags:** atp.Status=draft	
Available via	Mka.h	

⌋

8.3.11 Mka_GetMacSecStatisticsNotification

[CP_SWS_Mka_91024] Definition of callback function Mka_GetMacSecStatistics Notification

Status: DRAFT

⌈

Service Name	Mka_GetMacSecStatisticsNotification (draft)	
Syntax	`void Mka_GetMacSecStatisticsNotification ( uint8 MkaPaeIdx, Std_ReturnType Result )`	
Service ID [hex]	0x1e	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different MkaPaeIdx, Non reentrant for the same MkaPaeIdx	
Parameters (in)	MkaPaeIdx	Index of the PAE within the context of the MKA module
	Result	`E_OK`:MacSecStatistics have been received `E_NOT_OK`:MacSecStatistics have not been received
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	Callback to signal completion of call to EthIf_MacSecGetMacSecStatistics, EthIf_EthMacSec GetMacSecStatistics or EthIf_SwitchMacSecGetMacSecStatistics. **Tags:** atp.Status=draft	
Available via	Mka.h	

⌋

[CP_SWS_Mka_01006]

Status: DRAFT

⌈If the function `Mka_GetMacSecStatisticsNotification` is called with Result = E_OK, Mka shall assert that MacSec Statistics was successfully provided.⌋

[CP_SWS_Mka_01007]

Status: DRAFT

⌈If the function `Mka_GetMacSecStatisticsNotification` is called with Result = E_NOT_OK, Mka should retry providing MacSec Statistics.⌋

8.3.12 Mka_RxIndication

[CP_SWS_Mka_91029] Definition of callback function Mka_RxIndication

Status: DRAFT

⌈

Service Name	Mka_RxIndication (draft)	

Syntax	`void Mka_RxIndication (`
	`  PduIdType RxPduId,`
	`  const PduInfoType* PduInfoPtr`
	`)`	

Service ID [hex]	0x42	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different PduIds. Non reentrant for the same PduId.	
Parameters (in)	RxPduId	ID of the received PDU.
	PduInfoPtr	Contains the length (SduLength) of the received PDU, a pointer to a buffer (SduDataPtr) containing the PDU, and the MetaData related to this PDU.
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	Indication of a received PDU from a lower layer communication interface module. **Tags:** atp.Status=draft	
Available via	Mka.h	

⌋

8.3.13 Mka_TxConfirmation

[CP_SWS_Mka_91030] Definition of callback function Mka_TxConfirmation

Status: DRAFT

⌈

Service Name	Mka_TxConfirmation (draft)	

Syntax	`void Mka_TxConfirmation (`
	`  PduIdType TxPduId,`
	`  Std_ReturnType result`
	`)`	

Service ID [hex]	0x40	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different PduIds. Non reentrant for the same PduId.	
Parameters (in)	TxPduId	ID of the PDU that has been transmitted.
	result	`E_OK`: The PDU was transmitted successfully. `E_NOT_OK`: Transmission of the PDU failed.
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	

▽

△

Description	The lower layer communication interface module confirms the transmission of a PDU, or the failure to transmit a PDU. **Tags:** atp.Status=draft
Available via	Mka.h

⌋

8.3.14 Mka_MacSecUpdateSecYNotification

[CP_SWS_Mka_91031] Definition of callback function Mka_MacSecUpdateSecYNotification

Status: DRAFT

⌈

Service Name	Mka_MacSecUpdateSecYNotification (draft)	
Syntax	`void Mka_MacSecUpdateSecYNotification ( uint8 MkaPaeIdx, Std_ReturnType Result )`	
Service ID [hex]	0x21	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different MkaPaeIdx, Non reentrant for the same MkaPaeIdx	
Parameters (in)	MkaPaeIdx	Index of the PAE within the context of the MKA module
	Result	`E_OK`: TEthIf_MacSecUpdateSecY or EthIf_SwitchMacSec UpdateSecY has finished and SecY is updated with the provided parameters of EthIf_MacSecUpdateSecY or EthIf_SwitchMacSec UpdateSecY `E_NOT_OK`:SecY has not been updated with the provided parameters of EthIf_MacSecUpdateSecY or EthIf_SwitchMacSec UpdateSecY
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	Callback to signal completion of call to EthIf_MacSecUpdateSecY, EthIf_EthMacSecUpdateSec Y or EthIf_SwitchMacSecUpdateSecY has finished. **Tags:** atp.Status=draft	
Available via	Mka.h	

⌋

[CP_SWS_Mka_01008]

Status: DRAFT

⌈If the function `Mka_MacSecUpdateSecYNotification` is called with Result = E_OK, Mka shall assert that the MacSec Entity (SecY) is updated with the provided parameters of `EthIf_MacSecUpdateSecY` or `EthIf_SwitchMacSecUpdateSecY`.⌋

[CP_SWS_Mka_01009]

Status: DRAFT

⌈If the function `Mka_MacSecUpdateSecYNotification` is called with Result = E_NOT_OK, Mka should retry again `EthIf_MacSecUpdateSecY` or `EthIf_SwitchMacSecUpdateSecY`.⌋

8.3.15 Mka_MacSecAddTxSaNotification

[CP_SWS_Mka_91032] Definition of callback function Mka_MacSecAddTxSaNotification

Status: DRAFT

⌈

Service Name	Mka_MacSecAddTxSaNotification (draft)	
Syntax	`void Mka_MacSecAddTxSaNotification ( uint8 MkaPaeIdx, Std_ReturnType Result )`	
Service ID [hex]	0x22	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different MkaPaeIdx, Non reentrant for the same MkaPaeIdx	
Parameters (in)	MkaPaeIdx	Index of the PAE within the context of the MKA module
	Result	`E_OK`: E_OK: EthIf_MacSecAddTxSa or EthIf_SwitchMacSecAddTxSa has finished and Transmission Secure Association is created `E_NOT_OK`:The Transmission Secure Association is not created through EthIf_MacSecAddTxSa or EthIf_SwitchMacSecAddTxSa
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	Callback to signal completion of call to EthIf_MacSecAddTxSa, EthIf_EthMacSecAddTxSa or EthIf_SwitchMacSecAddTxSa. **Tags:** atp.Status=draft	
Available via	Mka.h	

⌋

[CP_SWS_Mka_01010]

Status: DRAFT

⌈If the function `Mka_MacSecAddTxSaNotification` is called with Result = E_OK, Mka shall assert that the Transmission Secure Association has been created in the Transceiver through `EthIf_MacSecAddTxSa` or `EthIf_SwitchMacSecAddTxSa`.⌋

[CP_SWS_Mka_01011]

Status: DRAFT

⌈If the function `Mka_MacSecAddTxSaNotification` is called with Result = E_NOT_OK, Mka should retry again `EthIf_MacSecAddTxSa` or `EthIf_SwitchMacSecAddTxSa`.⌋

8.3.16 Mka_MacSecAddRxSaNotification

[CP_SWS_Mka_91033] Definition of callback function Mka_MacSecAddRxSaNotification

Status: DRAFT

⌈

Service Name	Mka_MacSecAddRxSaNotification (draft)	
Syntax	`void Mka_MacSecAddRxSaNotification ( uint8 MkaPaeIdx, Std_ReturnType Result )`	
Service ID [hex]	0x23	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different MkaPaeIdx, Non reentrant for the same MkaPaeIdx	
Parameters (in)	MkaPaeIdx	Index of the PAE within the context of the MKA module
	Result	`E_OK`: EthIf_MacSecAddRxSa or EthIf_SwitchMacSecAddRxSa has finished and Reception Secure Association is created `E_NOT_OK`:The Reception Secure Association is not created through EthIf_MacSecAddRxSa or EthIf_SwitchMacSecAddRxSa
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	Callback to signal completion of call to EthIf_MacSecAddRxSa, EthIf_EthMacSecAddRxSa or EthIf_SwitchMacSecAddRxSa has finished. **Tags:** atp.Status=draft	
Available via	Mka.h	

⌋

[CP_SWS_Mka_01012]

Status: DRAFT

⌈If the function `Mka_MacSecAddRxSaNotification` is called with Result = E_OK, Mka shall assert that the Reception Secure Association has been created in the Transceiver through `EthIf_MacSecAddRxSa` or `EthIf_SwitchMacSecAddRxSa`.⌋

[CP_SWS_Mka_01013]

Status: DRAFT

⌈If the function `Mka_MacSecAddRxSaNotification` is called with Result = E_NOT_OK, Mka should retry again `EthIf_MacSecAddRxSa` or `EthIf_SwitchMacSecAddRxSa`.⌋

8.3.17 Mka_CanSecUpdateSecYNotification

[SWS_Mka_91044] Definition of callback function Mka_CanSecUpdateSecYNotification ⌈

Service Name	Mka_CanSecUpdateSecYNotification	
Syntax	`void Mka_CanSecUpdateSecYNotification ( uint8 MkaPaeIdx, uint8 SecYIdx, Std_ReturnType Result )`	
Service ID [hex]	0x25	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different MkaPaeIdx, Non reentrant for the same MkaPaeIdx	
Parameters (in)	MkaPaeIdx	Index of the PAE within the context of the MKA module
	SecYIdx	SecY that has been updated
	Result	`E_OK`: CanIf_CanSecUpdateSecY has finished and SecY is updated with the provided parameters of CanIf_CanSecUpdate SecY `E_NOT_OK`: SecY has not been updated with the provided parameters of CanIf_CanSecUpdateSecY.
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	Callback to to notify that CanIf_CanSecUpdateSecY finished. Note: This API is preliminary and must be implemented by integration code to complete the CANsec control plane. Functional specification will be added once the CANsec key agreement specification is released by CiA.	
Available via	Mka.h	

⌋

8.3.18 Mka_CanSecAddTxSaNotification

[SWS_Mka_91045] Definition of callback function Mka_CanSecAddTxSaNotification ⌈

Service Name	Mka_CanSecAddTxSaNotification	
Syntax	`void Mka_CanSecAddTxSaNotification ( uint8 MkaPaeIdx, uint8 SecYIdx, uint16 Csci, uint8 An, Std_ReturnType Result )`	
Service ID [hex]	0x26	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different MkaPaeIdx, Non reentrant for the same MkaPaeIdx	
Parameters (in)	MkaPaeIdx	Index of the PAE within the context of the MKA module
	SecYIdx	SecY for which a Tx SA has been added
	Csci	CANsec Secure Channel Identifier for which the SA has been added

▽

△

	An	Association Number used in the CANsec's transmission secure association
	Result	E_OK: CanIf_CanSecAddTxSa has finished and Transmission Secure Association is created E_NOT_OK: The Transmission Secure Association is not created through CanIf_CanSecAddTxSa.
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	Callback to notify that CanIf_CanSecAddTxSa has finished. Note: This API is preliminary and must be implemented by integration code to complete the CANsec control plane. Functional specification will be added once the CANsec key agreement specification is released by CiA.	
Available via	Mka.h	

⌋

8.3.19 Mka_CanSecAddRxSaNotification

[SWS_Mka_91046] Definition of callback function Mka_CanSecAddRxSaNotification ⌈

Service Name	Mka_CanSecAddRxSaNotification	

Syntax	`void Mka_CanSecAddRxSaNotification (`
	`  uint8 MkaPaeIdx,`
	`  uint8 SecYIdx,`
	`  uint16 Csci,`
	`  uint8 An,`
	`  Std_ReturnType Result`
	`)`	

Service ID [hex]	0x27	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different MkaPaeIdx, Non reentrant for the same MkaPaeIdx	
Parameters (in)	MkaPaeIdx	Index of the PAE within the context of the MKA module
	SecYIdx	SecY for which a Rx SA has been added
	Csci	CANsec Secure Channel Identifier for which the SA has been added
	An	Association Number used in the CANsec's reception secure association
	Result	E_OK: CanIf_CanSecAddRxSa has finished and Reception Secure Association is created E_NOT_OK: The Reception Secure Association is not created through CanIf_CanSecAddRxSa.
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	Callback to notify that CanIf_CanSecAddRxSa has finished. Note: This API is preliminary and must be implemented by integration code to complete the CANsec control plane. Functional specification will be added once the CANsec key agreement specification is released by CiA.	
Available via	Mka.h	

⌋

8.3.20 Mka_GetCanSecSecYStatisticsNotification

[SWS_Mka_91047] Definition of callback function Mka_GetCanSecSecYStatistics Notification ⌈

Service Name	Mka_GetCanSecSecYStatisticsNotification	
Syntax	`void Mka_GetCanSecSecYStatisticsNotification ( uint8 MkaPaeIdx, uint8 SecYIdx, Std_ReturnType Result )`	
Service ID [hex]	0x28	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different MkaPaeIdx, Non reentrant for the same MkaPaeIdx	
Parameters (in)	MkaPaeIdx	Index of the PAE within the context of the MKA module
	SecYIdx	SecY for which the statistics have been requested
	Result	`E_OK`: CanSecStatistics have been received `E_NOT_OK`: CanSecStatistics have not been received
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	Callback notify that Can_GetCanSecSecYStatistics has finished and provides the requested statistics. Note: This API is preliminary and must be implemented by integration code to complete the CANsec control plane. Functional specification will be added once the CANsec key agreement specification is released by CiA.	
Available via	Mka.h	

⌋

8.3.21 Mka_GetCanSecTxScStatisticsNotification

[SWS_Mka_91048] Definition of callback function Mka_GetCanSecTxScStatistics Notification ⌈

Service Name	Mka_GetCanSecTxScStatisticsNotification	
Syntax	`void Mka_GetCanSecTxScStatisticsNotification ( uint8 MkaPaeIdx, uint8 SecYIdx, uint16 Csci, Std_ReturnType Result )`	
Service ID [hex]	0x29	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different MkaPaeIdx, Non reentrant for the same MkaPaeIdx	
Parameters (in)	MkaPaeIdx	Index of the PAE within the context of the MKA module
	SecYIdx	SecY for which the statistics have been requested
	Csci	CANsec Secure Channel Identifier of a transmission Secure Channel for which the statistics have been requested

▽

△

	Result	E_OK: CanSecStatistics have been received E_NOT_OK: CanSecStatistics have not been received
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	Callback to notify that CanIf_CanSecGetCanSecTxScStatistics has finished and provides the requested statistics. Note: This API is preliminary and must be implemented by integration code to complete the CANsec control plane. Functional specification will be added once the CANsec key agreement specification is released by CiA.	
Available via	Mka.h	

⌋

8.3.22 Mka_GetCanSecRxScStatisticsNotification

[SWS_Mka_91049] Definition of callback function Mka_GetCanSecRxScStatisticsNotification ⌈

Service Name	Mka_GetCanSecRxScStatisticsNotification	

Syntax	`void Mka_GetCanSecRxScStatisticsNotification (`
	`  uint8 MkaPaeIdx,`
	`  uint8 SecYIdx,`
	`  uint16 Csci,`
	`  Std_ReturnType Result`
	`)`	

Service ID [hex]	0x2a	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different MkaPaeIdx, Non reentrant for the same MkaPaeIdx	
Parameters (in)	MkaPaeIdx	Index of the PAE within the context of the MKA module
	SecYIdx	SecY for which the statistics have been requested
	Csci	CANsec Secure Channel Identifier of a reception Secure Channel for which the statistics have been requested
	Result	E_OK: CanSecStatistics have been received E_NOT_OK: CanSecStatistics have not been received
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	Callback to notify that CanIf_CanSecGetCanSecRxScStatistics has finished and provides the requested statistics. Note: This API is preliminary and must be implemented by integration code to complete the CANsec control plane. Functional specification will be added once the CANsec key agreement specification is released by CiA.	
Available via	Mka.h	

⌋

8.3.23 Mka_CanControllerModeIndication

[SWS_Mka_91050] Definition of callback function Mka_CanControllerModeIndication ⌈

Service Name	Mka_CanControllerModeIndication	

Syntax	`void Mka_CanControllerModeIndication (`
	`  uint8 MkaPaeIdx,`
	`  Can_ControllerStateType ControllerMode`
	`)`	

Service ID [hex]	0x2b	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different MkaPaeIdx, Non reentrant for the same MkaPaeIdx	
Parameters (in)	MkaPaeIdx	Index of the PAE within the context of the MKA module
	ControllerMode	Mode to which the CAN controller transitioned
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	Callback to indicate a CAN controller state transition. Note: This API is non-functional and serves the purpose to complete the CANsec control plane APIs. Functional specification will be added once the CANsec key agreement specification is released by CiA.	
Available via	Mka.h	

⌋

8.4 Scheduled functions

These functions are directly called by Basic Software Scheduler. The following functions shall have no return value and no parameter. All functions shall be non reentrant.

8.4.1 Mka_MainFunction

[CP_SWS_Mka_91034] Definition of scheduled function Mka_MainFunction

Status: DRAFT

⌈

Service Name	Mka_MainFunction (draft)

Syntax	`void Mka_MainFunction (`
	`  void`
	`)`

Service ID [hex]	0x24
Description	Main function for cyclic call. **Tags:** atp.Status=draft
Available via	Mka.h

⌋

[CP_SWS_Mka_01005]

Status: DRAFT

⌈The frequency of invocations of `Mka_MainFunction` is determined by the configuration parameter `MkaMainFunctionPeriod`.⌋

8.5 Expected interfaces

In this chapter all interfaces required from other modules are listed.

8.5.1 Mandatory interfaces

Note: This section defines all interfaces, which are required to fulfill the core functionality of the module.

[CP_SWS_Mka_91006] Definition of mandatory interfaces required by module Mka

Status: DRAFT

⌈

API Function	*Header File*	*Description*
Csm_Encrypt	Csm.h	Encrypts the given data and store the ciphertext in the memory location pointed by the result pointer.
Csm_KeyDerive	Csm.h	Derives a new key by using the key elements in the given key identified by the keyId. The given key contains the key elements for the password and salt. The derived key is stored in the key element with the id 1 of the key identified by targetCryptoKeyId.
Csm_KeyElementGet	Csm.h	Retrieves the key element bytes from a specific key element of the key identified by the keyId and stores the key element in the memory location pointed by the key pointer.
Csm_KeyElementSet	Csm.h	Sets the given key element bytes to the key identified by keyId.
Csm_KeySetValid	Csm.h	Sets the key state of the key identified by keyId to valid.
Csm_MacGenerate	Csm.h	Uses the given data to perform a MAC generation and stores the MAC in the memory location pointed to by the MAC pointer.
Csm_MacVerify	Csm.h	Verifies the given MAC by comparing if the MAC is generated with the given data.
Csm_RandomGenerate	Csm.h	Generate a random number and stores it in the memory location pointed by the result pointer.
EthIf_GetPhysAddr	EthIf.h	Obtains the physical source address used by the indexed controller

▽

△

API Function	Header File	Description
EthIf_UpdatePhysAddrFilter	EthIf.h	Update the physical source address to/from the indexed controller filter. If the Ethernet Controller is not capable to do the filtering or managing same multicast address of several virtual controllers, the Ethernet Controller software has to handle it.
LSduR_MkaTransmit (draft)	LSduR_<module>.h	Requests transmission of a PDU.
NvM_EraseNvBlock	NvM.h	Service to erase a NV block.
NvM_ReadBlock	NvM.h	Service to copy the data of the NV block to its corresponding RAM block.
NvM_WriteBlock	NvM.h	Service to copy the data of the RAM block to its corresponding NV block.

⌋

8.5.2 Optional interfaces

This section defines all interfaces, which are required to fulfill an optional functionality of the module.

[CP_SWS_Mka_91007] Definition of optional interfaces requested by module Mka

Status: DRAFT

⌈

API Function	Header File	Description
Csm_Decrypt	Csm.h	Decrypts the given encrypted data and store the decrypted plaintext in the memory location pointed by the result pointer.
Csm_Hash	Csm.h	Uses the given data to perform the hash calculation and stores the hash.
Csm_JobKeyUnwrap	Csm.h	Unwraps the wrapped key given by the ciphertextPtr with the key wrapping key associated with the jobId. The unwrapped key will be written to targetKeyId in the element with the Id 1. If an authentication shall be done, the authenticator shall be referenced with authenticatorPtr. If the algorithm has no authenticator or the algorithm has an authenticator and no authentication shall be done, authenticator LengthPtr shall be set to zero. If the algorithm has no authentication, verifyPtr will be set to CRYPTO_E_VER_OK.
Csm_JobKeyWrap	Csm.h	Wraps the plaintext key given by sourceKeyId (in the key element with the id 1) with the key wrapping key associated with the jobId. The wrapped key will be written to ciphertextPtr. If the algorithm does provide an authenticator this will be written to authenticator Ptr. If the algorithm has no authenticator or the algorithm has an authenticator and no authentication shall be done, authenticatorLengthPtr shall be set to zero.

▽

△

API Function	*Header File*	*Description*
Dem_SetEventStatus	Dem.h	Called by SW-Cs or BSW modules to report monitor status information to the Dem. BSW modules calling Dem_SetEventStatus can safely ignore the return value. This API will be available only if ({Dem/Dem ConfigSet/DemEventParameter/DemEvent ReportingType} == STANDARD_REPORTING)
Det_ReportError	Det.h	Service to report development errors.
Det_ReportRuntimeError	Det.h	Service to report runtime errors. If a callout has been configured then this callout shall be called.
EthIf_MacSecAddRxSa (draft)	EthIf.h	Request the Ethernet Inferface (MACsec per SW) or the Ethernet Transceiver Driver to create a Reception Secure Association in the Transceiver. The Short Secure Channel Identifier is included to support XPN configurations. **Tags:** atp.Status=draft
EthIf_MacSecAddTxSa (draft)	EthIf.h	Requests the Ethernet Inferface (MACsec per SW) or the Ethernet Transceiver Driver to create a Transmission Secure Association in the Transceiver. The Short Secure Channel Identifier is included to support XPN configurations. **Tags:** atp.Status=draft
EthIf_MacSecDeleteRxSa (draft)	EthIf.h	Request the Ethernet Inferface (MACsec per SW) or the Ethernet Transceiver Driver to remove the Reception Secure Association identified by the provided Association Number. **Tags:** atp.Status=draft
EthIf_MacSecDeleteTxSa (draft)	EthIf.h	Request the Ethernet Inferface (MACsec per SW) or the Ethernet Transceiver Driver to remove the Transmission Secure Association identified by the provided Association Number. **Tags:** atp.Status=draft
EthIf_MacSecGetMacSecStatistics (draft)	EthIf.h	Request the Ethernet Inferface (MACsec per SW) or the Ethernet Transceiver Driver to provide MACsec statistics. The result is returned through EthIf_Mac SecGetMacSecStatisticsNotification **Tags:** atp.Status=draft
EthIf_MacSecGetTxSaNextPn (draft)	EthIf.h	Request the Ethernet Inferface (MACsec per SW) or the Ethernet Transceiver Driver to return the Packet Number that is used for the next packet in the given Transmission Secure Association. **Tags:** atp.Status=draft
EthIf_MacSecInitRxSc (draft)	EthIf.h	Requests the Ethernet Inferface (MACsec per SW) or the Ethernet Transceiver Driver to configure a Reception Secure Channel for the given Secure Channel Identifier. **Tags:** atp.Status=draft
EthIf_MacSecOperational (draft)	EthIf.h	To inform EthIf that MacSec is operational and that EthSM can be informed. (Ethernet Inferface (MACsec per SW) or the Ethernet Transceiver Driver) **Tags:** atp.Status=draft

▽

△

API Function	*Header File*	*Description*
EthIf_MacSecResetRxSc (draft)	EthIf.h	Requests the Ethernet Inferface (MACsec per SW) or the Ethernet Transceiver Driver to delete the Reception Secure Channel addressed with the given Reception Secure Channel Identifier. **Tags:** atp.Status=draft
EthIf_MacSecSetControlledPort Enabled (draft)	EthIf.h	Requests to set the Controlled Port enabled parameter of a PAE. **Tags:** atp.Status=draft
EthIf_MacSecUpdateRxSa (draft)	EthIf.h	Request the Ethernet Inferface (MACsec per SW) or the Ethernet Transceiver Driver to update the Reception Secure Association with the given Packet Number. The Active parameter is included to change the specified AN status. **Tags:** atp.Status=draft
EthIf_MacSecUpdateSecY (draft)	EthIf.h	Requests the Ethernet Inferface (MACsec per SW) or the Ethernet Transceiver to update the SecY/PAC of the PHY with the provided parameters. A Transmission Secure Channel with the provided SCI shall be configured during the first call. A pointer to a MACsec Basic Parameters Configuration file shall be provided to create the Secure Channel. **Tags:** atp.Status=draft
EthIf_MacSecUpdateTxSa (draft)	EthIf.h	Requests the Ethernet Inferface (MACsec per SW) or the Ethernet Transceiver Driver to update the Transmission Secure Association with the given Packet Number. The Active parameter is included to change the specified AN status. **Tags:** atp.Status=draft
EthIf_SetSwitchMgmtInfo	EthIf.h	Provides additional management information along to an Ethernet frame that requires special treatment within the Switch. For direct data provision, it has to be called before the transmit request is called. For indirect data provision, it can also be called in the context of the TriggerTransmit API.
EthIf_SwitchMacSecAddTxSa (draft)	EthIf.h	Requests the Ethernet Switch Driver to create a Transmission Secure Association in the provided port. The Short Secure Channel Identifier is included to support XPN configurations. **Tags:** atp.Status=draft
EthIf_SwitchMacSecDeleteRxSa (draft)	EthIf.h	Request the Ethernet Switch Driver to remove the Reception Secure Association identified by the provided Association Number. **Tags:** atp.Status=draft
EthIf_SwitchMacSecDeleteTxSa (draft)	EthIf.h	Request the Ethernet Switch Driver to remove the Transmission Secure Association identified by the provided Association Number. **Tags:** atp.Status=draft
EthIf_SwitchMacSecGetMacSec Statistics (draft)	EthIf.h	Request the Ethernet switch Driver to provide MACsec statistics. The result is returned throughEth If_SwitchMacSecGetMacSecStatisticsNotification **Tags:** atp.Status=draft

▽

△

API Function	*Header File*	*Description*
EthIf_SwitchMacSecGetTxSaNextPn (draft)	EthIf.h	Request the Ethernet Switch Driver to return the Packet Number that is used for the next packet in the given Transmission Secure Association. **Tags:** atp.Status=draft
EthIf_SwitchMacSecInitRxSc (draft)	EthIf.h	Requests the Ethernet Switch Driver to configure a Reception Secure Channel for the given Secure Channel Identifier. **Tags:** atp.Status=draft
EthIf_SwitchMacSecOperational (draft)	EthIf.h	To inform EthIf that MacSec is operational and that EthSM can be notified. (Switch case) **Tags:** atp.Status=draft
EthIf_SwitchMacSecResetRxSc (draft)	EthIf.h	Requests the Ethernet Switch Driver to delete the Reception Secure Channel addressed with the given Reception Secure Channel Identifier. **Tags:** atp.Status=draft
EthIf_SwitchMacSecSetControlledPort Enabled (draft)	EthIf.h	Requests to set the Controlled Port enabled parameter of a PAE. **Tags:** atp.Status=draft
EthIf_SwitchMacSecUpdateRxSa (draft)	EthIf.h	Request the Ethernet Switch Driver to update the Reception Secure Association with the given Packet Number. The Active parameter is included to change the specified AN status. **Tags:** atp.Status=draft
EthIf_SwitchMacSecUpdateSecY (draft)	EthIf.h	Requests the Ethernet Switch to update the SecY/ PAC of the the provided port with the provided parameters. A Transmission Secure Channel with the provided SCI shall be configured during the first call. A pointer to a MACsec Basic Parameters Configuration file shall be provided to create the Secure Channel. **Tags:** atp.Status=draft
EthIf_SwitchMacSecUpdateTxSa (draft)	EthIf.h	Requests the Ethernet Switch Driver to update the Transmission Secure Association with the given Packet Number. The Active parameter is included to change the specified AN status. **Tags:** atp.Status=draft
IdsM_ReportSecurityEvent	IdsM.h	This API is the interface to report security events to the IdsM.

⌋

8.5.3 Configurable interfaces

In this section, all interfaces are listed where the target function could be configured. The target function is usually a callback function. The names of this kind of interfaces are not fixed because they are configurable.

[CP_SWS_Mka_91036] Definition of configurable interface <User_GetMacSec StatisticsCallbackNotification> ⌈

Service Name	<User_GetMacSecStatisticsCallbackNotification>	
Syntax	`void <User_GetMacSecStatisticsCallbackNotification> ( uint8 MkaPaeIdx )`	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different MkaPaeIdx, Non reentrant for the same MkaPaeIdx	
Parameters (in)	MkaPaeIdx	Index of the PAE within the context of the MKA module
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	Indicates and provides MacSecStatistics which were requested through Mka_GetMacSec Statistics [CP_SWS_Mka_91019]	
Available via	Mka_Externals.h	

⌋

8.6 Service Interfaces

There are no service interfaces defined.

9 Sequence diagrams

The sequence diagrams show the basic operations carried out during operation.
The purpose of the sequence diagrams is to depict the expected behavior of the communication stack at a glance.

The communication initialization sequence diagrams illustrate how the MKA module gets involved in the Ethernet stack start-up including upper and lower layer modules.

9.1 Communication initialization with MACsec

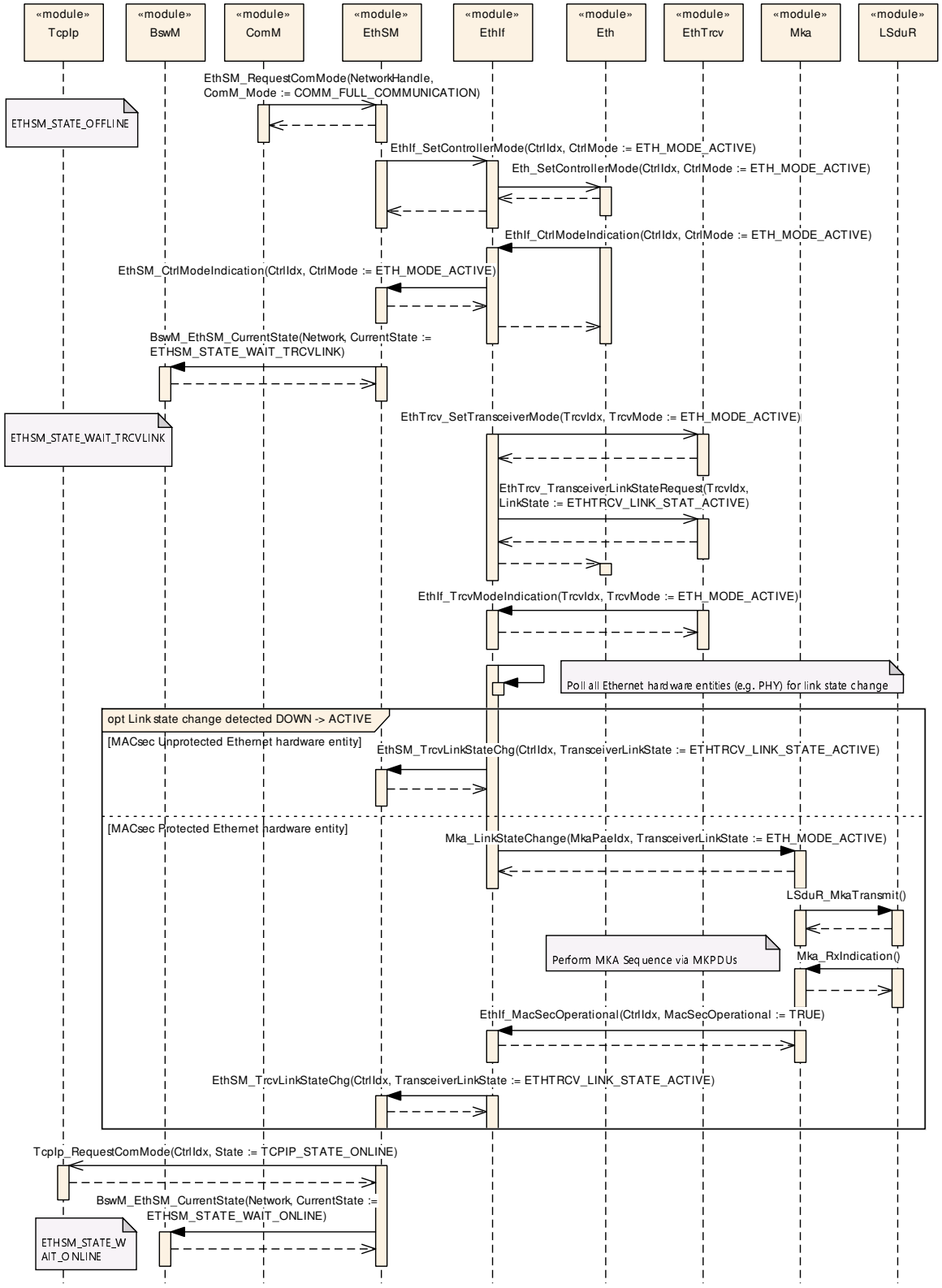

Figure 9.1: Communication initialization with MACsec protected EthIf

9.2 Communication initialization with MACsec and Switch

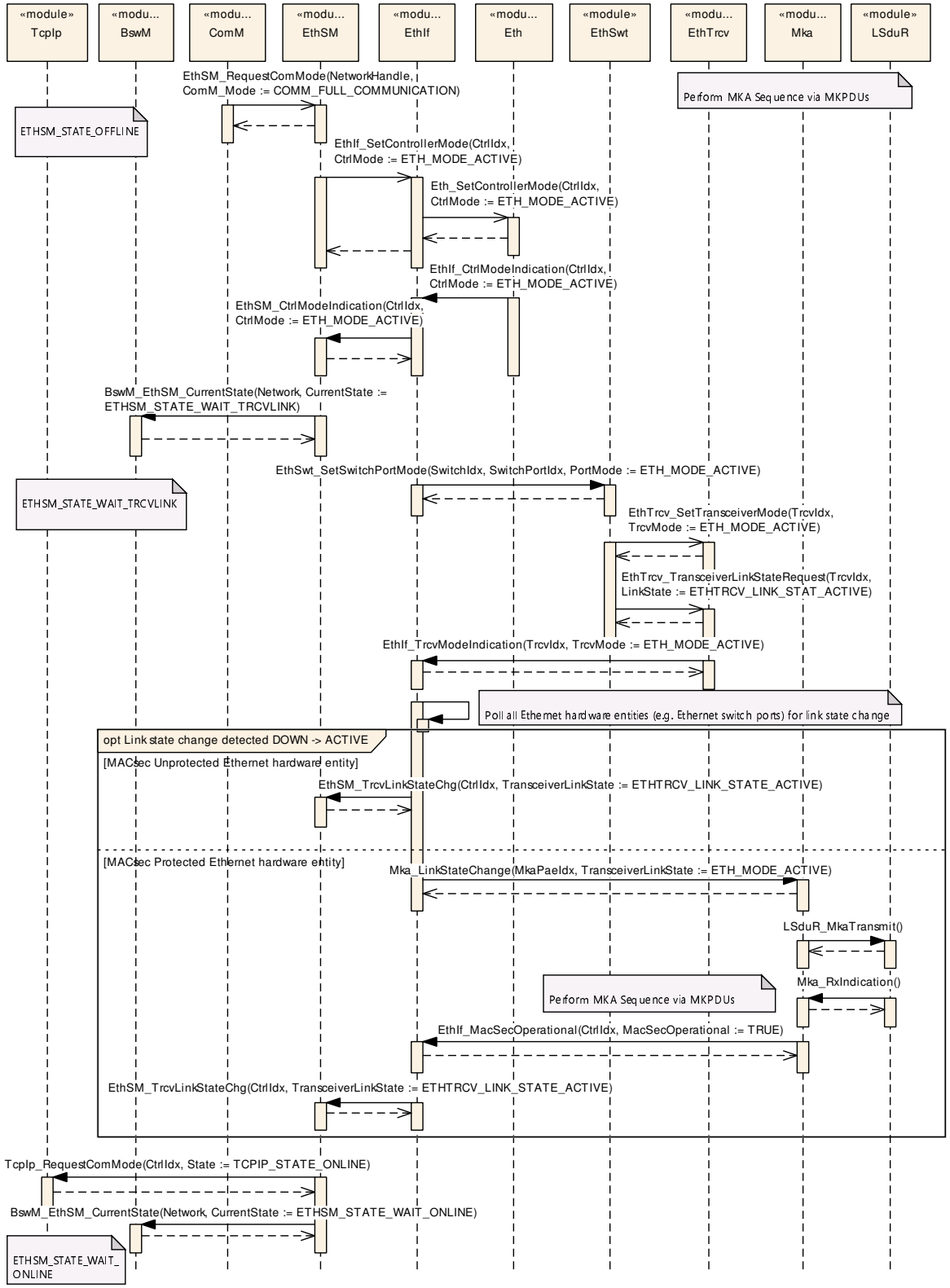

Figure 9.2: Communication initialization with MACsec protected EthIf and Switch

9.3 MKA exchange - KeyServer

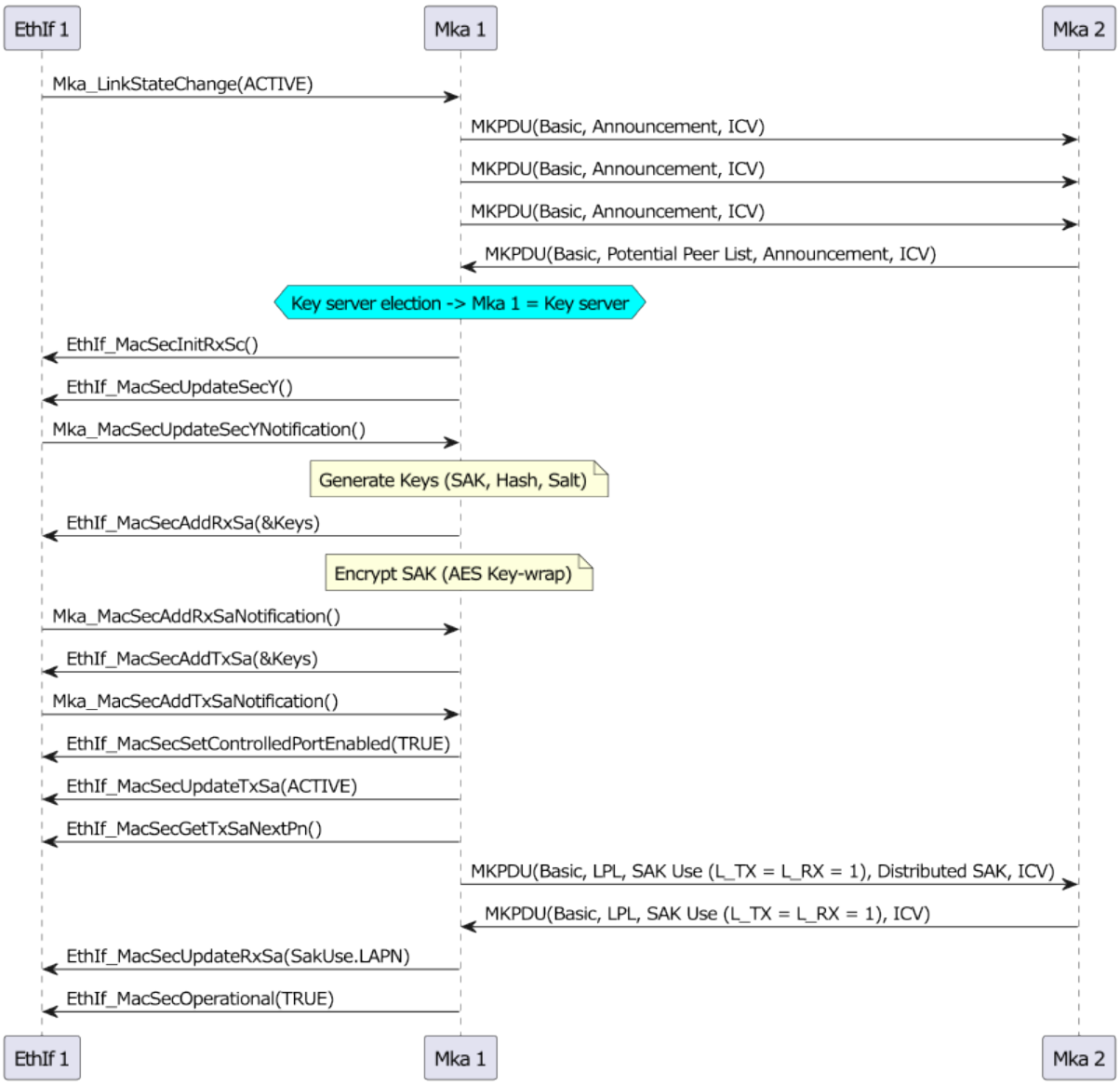

Figure 9.3: MKA exchange from the view of a Key Server

9.4 MKA exchange - Peer

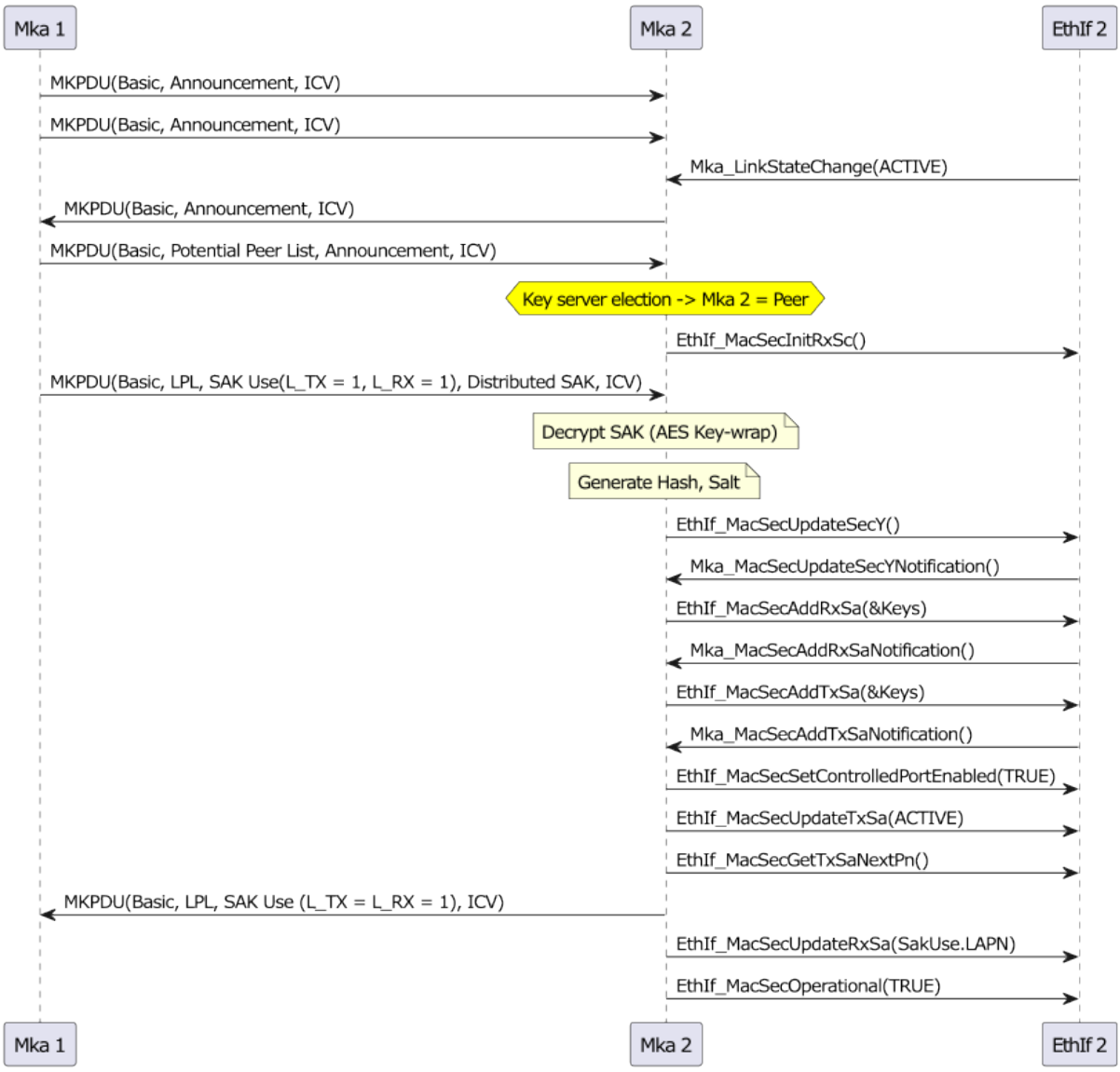

Figure 9.4: MKA exchange from the view of a Peer

10 Configuration specification

In general, this chapter defines configuration parameters and their clustering into containers. In order to support the specification Chapter 10.1 describes fundamentals. It also specifies a template (table) you shall use for the parameter specification. We intend to leave Chapter 10.1 in the specification to guarantee comprehension.

Chapter 10.2 specifies the structure (containers) and the parameters of the module MKA.

Chapter 10.3 specifies published information of the module MKA.

10.1 How to read this chapter

For details refer to [2] Chapter 10.1 *"Introduction to configuration specification"*.

10.2 Containers and configuration parameters

The following chapters summarize all configuration parameters. The detailed meanings of the parameters describe Chapter 7 and Chapter 8.

10.2.1 Mka

[ECUC_Mka_00001] Definition of EcucModuleDef Mka

Status: DRAFT

⌈

Module Name	Mka
Description	Configuration of the MACsec Key Agreement module.
Post-Build Selectable Support	false
Post-Build Loadable Support	false

Included Containers		
Container Name	**Multiplicity**	**Description**
MkaCryptoAlgoConfig	1..255	Cryptography configuration for MACsec. **Tags:** atp.Status=draft
MkaGeneral	1	This container holds the general parameters of the MKA protocol which apply to ports that are referencing this container. **Tags:** atp.Status=draft
MkaPaeConfiguration	1..255	Common MKA configuration for a PAE. **Tags:** atp.Status=draft
MkaPaeInstance	1..255	MKA configuration of a controlled port. **Tags:** atp.Status=draft

⌋

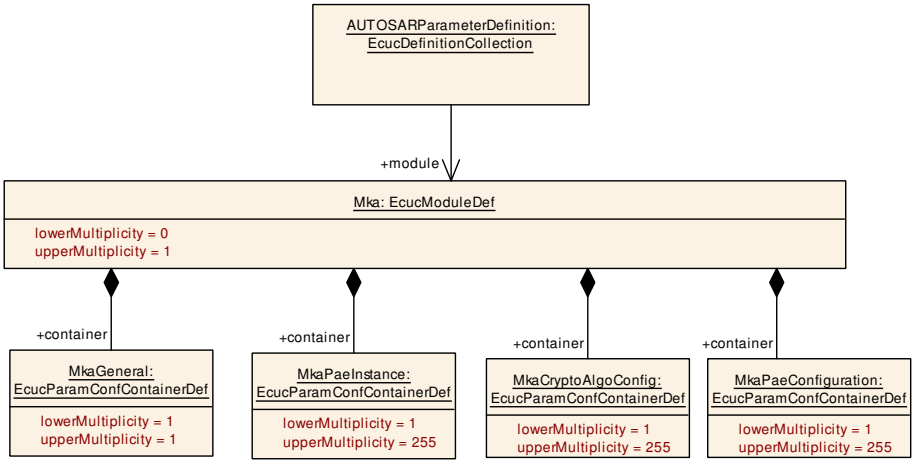

Figure 10.1: Mka

10.2.2 MkaGeneral

[ECUC_Mka_00002] Definition of EcucParamConfContainerDef MkaGeneral

Status: DRAFT

⌈

Container Name	MkaGeneral
Parent Container	Mka
Upstream mapping defined	–
Description	This container holds the general parameters of the MKA protocol which apply to ports that are referencing this container. **Tags:** atp.Status=draft
Multiplicity	1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
MkaDevErrorDetect	1	[ECUC_Mka_00034]
MkaEnableSecurityEventReporting	1	[ECUC_Mka_00061]
MkaHelloTime	1	[ECUC_Mka_00007]
MkaLifeTime	1	[ECUC_Mka_00009]
MkaMainFunctionPeriod	1	[ECUC_Mka_00035]
MkaSakRetireTime	1	[ECUC_Mka_00010]

Included Containers		
Container Name	**Multiplicity**	**Description**
MkaSecurityEventRefs	0..1	Container for the references to IdsMEvent elements representing the security events that the Mka module shall report to the IdsM in case the corresponding security related event occurs (and if MkaEnableSecurityEventReporting is set to "true"). The standardized security events in this container can be extended by vendor-specific security events. **Tags:** atp.Status=draft

⌋

[ECUC_Mka_00034] Definition of EcucBooleanParamDef MkaDevErrorDetect

Status: DRAFT

⌈

Parameter Name	MkaDevErrorDetect
Parent Container	MkaGeneral
Upstream mapping defined	–
Description	Switches the development error detection and notification on or off. - true: detection and notification is enabled. . false: detection and notification is disabled. **Tags:** atp.Status=draft
Multiplicity	1
Type	EcucBooleanParamDef
Default value	false
Pre-Compile only	true

⌋

[ECUC_Mka_00061] Definition of EcucBooleanParamDef MkaEnableSecurityEventReporting

Status: DRAFT

⌈

Parameter Name	MkaEnableSecurityEventReporting
Parent Container	MkaGeneral
Upstream mapping defined	–
Description	Switches the reporting of security events to the IdsM: - true: reporting is enabled. - false: reporting is disabled **Tags:** atp.Status=draft
Multiplicity	1
Type	EcucBooleanParamDef
Default value	false
Pre-Compile only	true

⌋

[ECUC_Mka_00007] Definition of EcucFloatParamDef MkaHelloTime

Status: DRAFT

⌈

Parameter Name	MkaHelloTime	
Parent Container	MkaGeneral	
Upstream mapping defined	–	
Description	Interval [s] between MKPDUs when two participants have an active MKA communication (the participants are included in the Live Peer list of each other). **Tags:** atp.Status=draft	
Multiplicity	1	
Type	EcucFloatParamDef	
Range	[0 .. INF]	
Default value	2	
Pre-Compile only	true	

⌋

[ECUC_Mka_00009] Definition of EcucFloatParamDef MkaLifeTime

Status: DRAFT

⌈

Parameter Name	MkaLifeTime	
Parent Container	MkaGeneral	
Upstream mapping defined	–	
Description	Time span [s] since last MKPDU was received from the other participant, to consider it alive. \newline In case no valid MKPDU from the other participant is received after Mka LifeTime, the Secure Channel is shut down. **Tags:** atp.Status=draft	
Multiplicity	1	
Type	EcucFloatParamDef	
Range	[0 .. INF]	
Default value	6	
Pre-Compile only	true	

⌋

[ECUC_Mka_00035] Definition of EcucFloatParamDef MkaMainFunctionPeriod

Status: DRAFT

⌈

Parameter Name	MkaMainFunctionPeriod
Parent Container	MkaGeneral
Upstream mapping defined	–
Description	The cycle time of the periodic main function of MKA. Defined in seconds. **Tags:** atp.Status=draft
Multiplicity	1
Type	EcucFloatParamDef

▽

△

Range	[0 .. INF]	
Default value	–	
Pre-Compile only	true	

⌋

[ECUC_Mka_00010] Definition of EcucFloatParamDef MkaSakRetireTime

Status: DRAFT

⌈

Parameter Name	MkaSakRetireTime	
Parent Container	MkaGeneral	
Upstream mapping defined	–	
Description	During an SAK rekey, time [s] to retire the previous SAK in use. **Tags:** atp.Status=draft	
Multiplicity	1	
Type	EcucFloatParamDef	
Range	[0 .. INF]	
Default value	3	
Pre-Compile only	true	

⌋

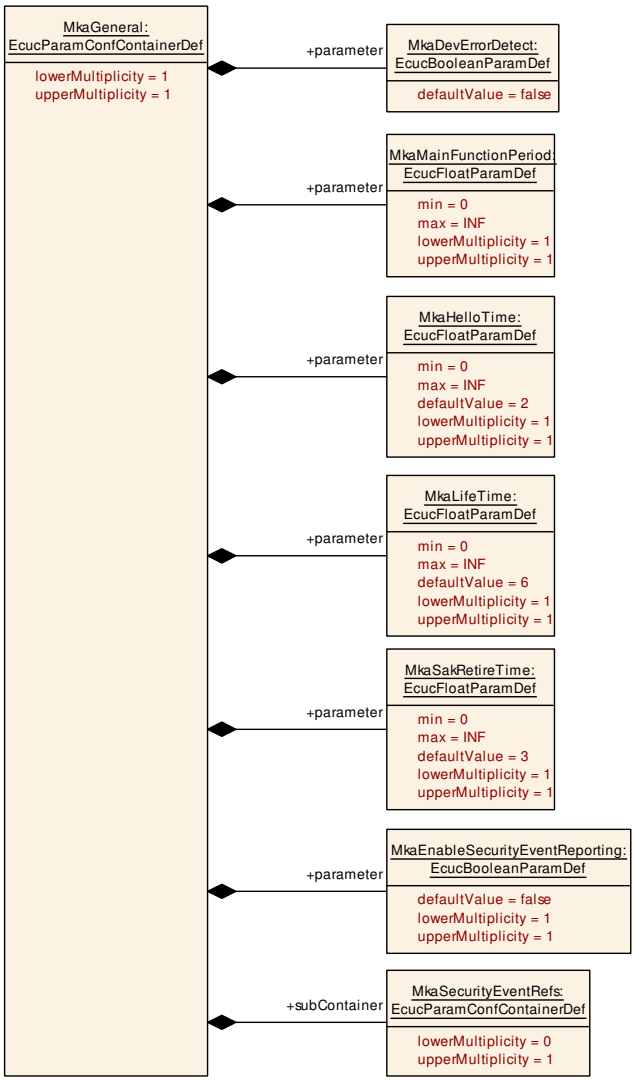

Figure 10.2: MkaGeneral

10.2.3 MkaPaeConfiguration

[ECUC_Mka_00033] Definition of EcucParamConfContainerDef MkaPaeConfiguration

Status: DRAFT

⌈

Container Name	MkaPaeConfiguration
Parent Container	Mka
Upstream mapping defined	–

▽

△

Description	Common MKA configuration for a PAE. **Tags:** atp.Status=draft
Multiplicity	1..255
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
MkaAutoStart	1	[ECUC_Mka_00012]
MkaCalculateSakHashKey	1	[ECUC_Mka_00077]
MkaGetMacSecStatisticsCallbackNotification	0..1	[ECUC_Mka_00076]
MkaPaeConfigurationIdx	1	[ECUC_Mka_00037]
MkaRetryBaseDelay	1	[ECUC_Mka_00004]
MkaRetryCyclicDelay	1	[ECUC_Mka_00005]
MkaSakRekeyTimeSpan	1	[ECUC_Mka_00024]

No Included Containers

⌋

[ECUC_Mka_00012] Definition of EcucBooleanParamDef MkaAutoStart

Status: DRAFT

⌈

Parameter Name	MkaAutoStart
Parent Container	MkaPaeConfiguration
Upstream mapping defined	[8, CP_TPS_SystemTemplate]
Description	Autostart or manual start of the PAE Instance. True := Autostart False := Manual Start If Autostart = False, the method Mka_StartPae is used to start the PAE instance. **Tags:** atp.Status=draft
Multiplicity	1
Type	EcucBooleanParamDef
Default value	true
Pre-Compile only	true

⌋

[ECUC_Mka_00077] Definition of EcucBooleanParamDef MkaCalculateSakHash Key ⌈

Parameter Name	MkaCalculateSakHashKey
Parent Container	MkaPaeConfiguration
Upstream mapping defined	[8, CP_TPS_SystemTemplate]

▽

△

Description	Indicates if Mka shall calculate and provide SAK hash key. • True = Mka shall calculate SAK hash key and provide it with HashKeyPtr of Mka_SakKeyPtrType. • False = Mka shall not calculate SAK hash key, HashKeyPtr of Mka_SakKeyPtrType shall be set to NULL pointer. (In this case SAK hash key is calculated by SecY itself.)
Multiplicity	1
Type	EcucBooleanParamDef
Default value	true
Pre-Compile only	true

⌋

[ECUC_Mka_00076] Definition of EcucFunctionNameDef MkaGetMacSecStatisticsCallbackNotification ⌈

Parameter Name	MkaGetMacSecStatisticsCallbackNotification
Parent Container	MkaPaeConfiguration
Upstream mapping defined	–
Description	Defines the full function name for the <User>_GetMacSecStatisticsCallback Notification() callback.
Multiplicity	0..1
Type	EcucFunctionNameDef
Default value	–
Regular Expression	–
Pre-Compile only	true

⌋

[ECUC_Mka_00037] Definition of EcucIntegerParamDef MkaPaeConfigurationIdx

Status: DRAFT

⌈

Parameter Name	MkaPaeConfigurationIdx	
Parent Container	MkaPaeConfiguration	
Upstream mapping defined	–	
Description	Instance ID of the MkaPaeConfiguration. **Tags:** atp.Status=draft	
Multiplicity	1	
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)	
Range	0 .. 255	
Default value	–	
Pre-Compile only	true	

⌋

[ECUC_Mka_00004] Definition of EcucFloatParamDef MkaRetryBaseDelay

Status: DRAFT

⌈

Parameter Name	MkaRetryBaseDelay
Parent Container	MkaPaeConfiguration
Upstream mapping defined	–
Description	The base delay in seconds for the retry phase of MKA. The retry have an exponential back off delay (1x base delay, 2x base delay, 4x base delay, ...) until the retry delay overflows the MkaRetryCyclicDelay value. **Tags:** atp.Status=draft
Multiplicity	1
Type	EcucFloatParamDef
Range	[0 .. INF]
Default value	–
Pre-Compile only	true

⌋

[ECUC_Mka_00005] Definition of EcucFloatParamDef MkaRetryCyclicDelay

Status: DRAFT

⌈

Parameter Name	MkaRetryCyclicDelay
Parent Container	MkaPaeConfiguration
Upstream mapping defined	–
Description	Interval in seconds between retries after base delay with exponential back off. **Tags:** atp.Status=draft
Multiplicity	1
Type	EcucFloatParamDef
Range	[0 .. INF]
Default value	–
Pre-Compile only	true

⌋

[ECUC_Mka_00024] Definition of EcucFloatParamDef MkaSakRekeyTimeSpan

Status: DRAFT

⌈

Parameter Name	MkaSakRekeyTimeSpan
Parent Container	MkaPaeConfiguration
Upstream mapping defined	[8, CP_TPS_SystemTemplate]
Description	Time [s] to trigger the rekey of an in use SAK. If set to 0, the rekey will not be triggered after a time span. **Tags:** atp.Status=draft
Multiplicity	1

▽

△

Type	EcucFloatParamDef	
Range	[0 .. INF]	
Default value	–	
Pre-Compile only	true	

⌋

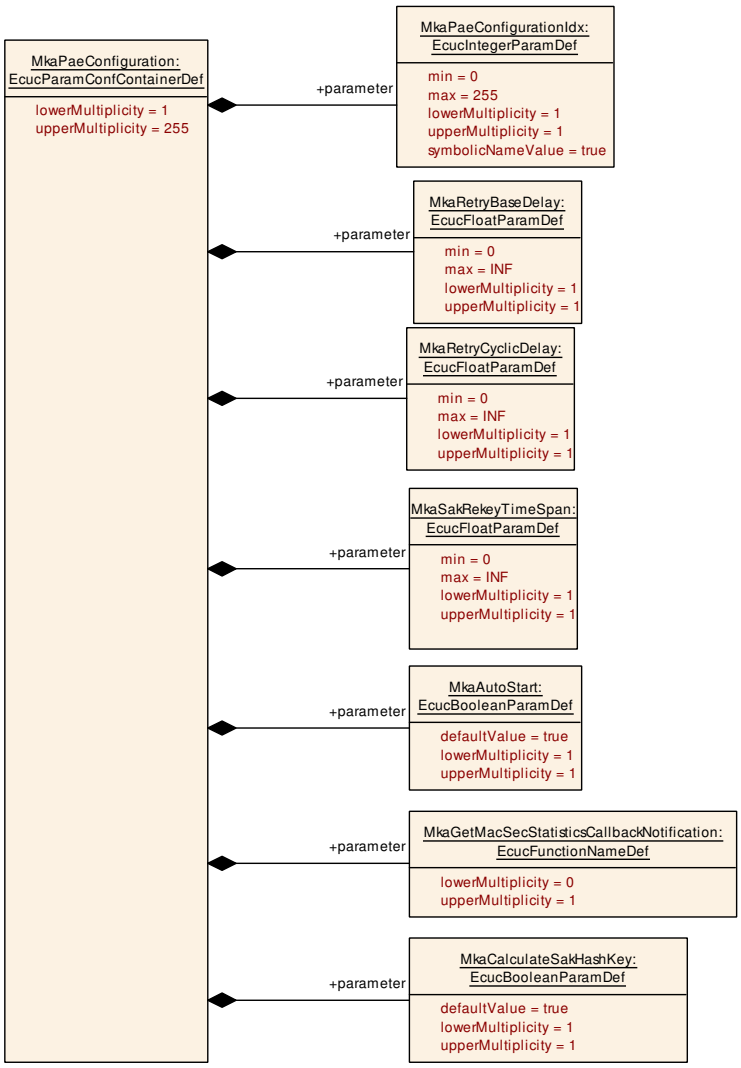

Figure 10.3: MkaPaeConfiguration

10.2.4 MkaCryptoAlgoConfig

[ECUC_Mka_00021] Definition of EcucParamConfContainerDef MkaCryptoAlgoConfig

Status: DRAFT

⌈

Container Name	MkaCryptoAlgoConfig
Parent Container	Mka
Upstream mapping defined	[8, CP_TPS_SystemTemplate]
Description	Cryptography configuration for MACsec. **Tags:** atp.Status=draft
Multiplicity	1..255
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
MkaCryptoAlgoConfigIdx	1	[ECUC_Mka_00053]
MkaMacSecCapability	1	[ECUC_Mka_00025]
MkaMacSecConfidentialityOffset	0..1	[ECUC_Mka_00026]
MkaMacSecReplayProtection	1	[ECUC_Mka_00027]
MkaMacSecReplayProtectionWindow	0..1	[ECUC_Mka_00028]

Included Containers		
Container Name	**Multiplicity**	**Description**
MkaCipherSuites	1..4	Cipher Suite configuration to use with MACsec. MkaCipherSuite Prio is present in case the MKA instance acts as a Key Server to select the cipher suite to use for MACsec.

⌋

[ECUC_Mka_00053] Definition of EcucIntegerParamDef MkaCryptoAlgoConfig Idx

Status: DRAFT

⌈

Parameter Name	MkaCryptoAlgoConfigIdx	
Parent Container	MkaCryptoAlgoConfig	
Upstream mapping defined	–	
Description	Instance ID of the configured Crypto configuration. **Tags:** atp.Status=draft	
Multiplicity	1	
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)	
Range	0 .. 255	
Default value	–	
Pre-Compile only	true	

⌋

[ECUC_Mka_00025] Definition of EcucEnumerationParamDef MkaMacSecCapability

Status: DRAFT

⌈

Parameter Name	MkaMacSecCapability	
Parent Container	MkaCryptoAlgoConfig	
Upstream mapping defined	[8, CP_TPS_SystemTemplate]	
Description	MACsec capability to use for MACsec. **Tags:** atp.Status=draft	
Multiplicity	1	
Type	EcucEnumerationParamDef	
Range	INTEGRITY_AND_ CONFIDENTIALITY	Protect integrity and confidentiality **Tags:** atp.Status=draft
	INTEGRITY_WITHOUT_ CONFIDENTIALITY	Only protect integrity, do not protect confidentiality **Tags:** atp.Status=draft
Default value	INTEGRITY_WITHOUT_CONFIDENTIALITY	
Pre-Compile only	true	

⌋

[ECUC_Mka_00026] Definition of EcucEnumerationParamDef MkaMacSecConfidentialityOffset

Status: DRAFT

⌈

Parameter Name	MkaMacSecConfidentialityOffset	
Parent Container	MkaCryptoAlgoConfig	
Upstream mapping defined	[8, CP_TPS_SystemTemplate]	
Description	The confidentiality Offset is only applicable if “Integrity and confidentiality” with a non-XPN cipher suite is selected. **Tags:** atp.Status=draft	
Multiplicity	0..1	
Type	EcucEnumerationParamDef	
Range	CONFIDENTIALITY_OFFSET_0	Start confidentiality protection at offset 0 (complete user data is encrypted) **Tags:** atp.Status=draft
	CONFIDENTIALITY_OFFSET_30	Start confidentiality protection at octet 30 (first 29 octets are not encrypted) **Tags:** atp.Status=draft
	CONFIDENTIALITY_OFFSET_50	Start confidentiality protection at octet 50 (first 49 octets are not encrypted) **Tags:** atp.Status=draft
Default value	CONFIDENTIALITY_OFFSET_0	
Pre-Compile only	true	

⌋

[ECUC_Mka_00027] Definition of EcucBooleanParamDef MkaMacSecReplayProtection

Status: DRAFT

⌈

Parameter Name	MkaMacSecReplayProtection
Parent Container	MkaCryptoAlgoConfig
Upstream mapping defined	[8, CP_TPS_SystemTemplate]
Description	MACsec replay protection parameter for MACsec.\newline The Replay Protection parameter is defined in the IEEE 802.1AE-2018 document, on chapter 10.4. It enables the replay protection if a packet is received with PacketNumber outside of the Window = PN - ProtectionWindow. **Tags:** atp.Status=draft
Multiplicity	1
Type	EcucBooleanParamDef
Default value	false
Pre-Compile only	true

⌋

[ECUC_Mka_00028] Definition of EcucIntegerParamDef MkaMacSecReplayProtectionWindow

Status: DRAFT

⌈

Parameter Name	MkaMacSecReplayProtectionWindow	
Parent Container	MkaCryptoAlgoConfig	
Upstream mapping defined	[8, CP_TPS_SystemTemplate]	
Description	In case replay protection is active, replay protection window.The Protection Window is a positive integer between 0 and 2^32-1 (No XPN) or 2^30-1 (XPN). **Tags:** atp.Status=draft	
Multiplicity	0..1	
Type	EcucIntegerParamDef	
Range	0 .. 18446744073709551615	
Default value	–	
Pre-Compile only	true	

⌋

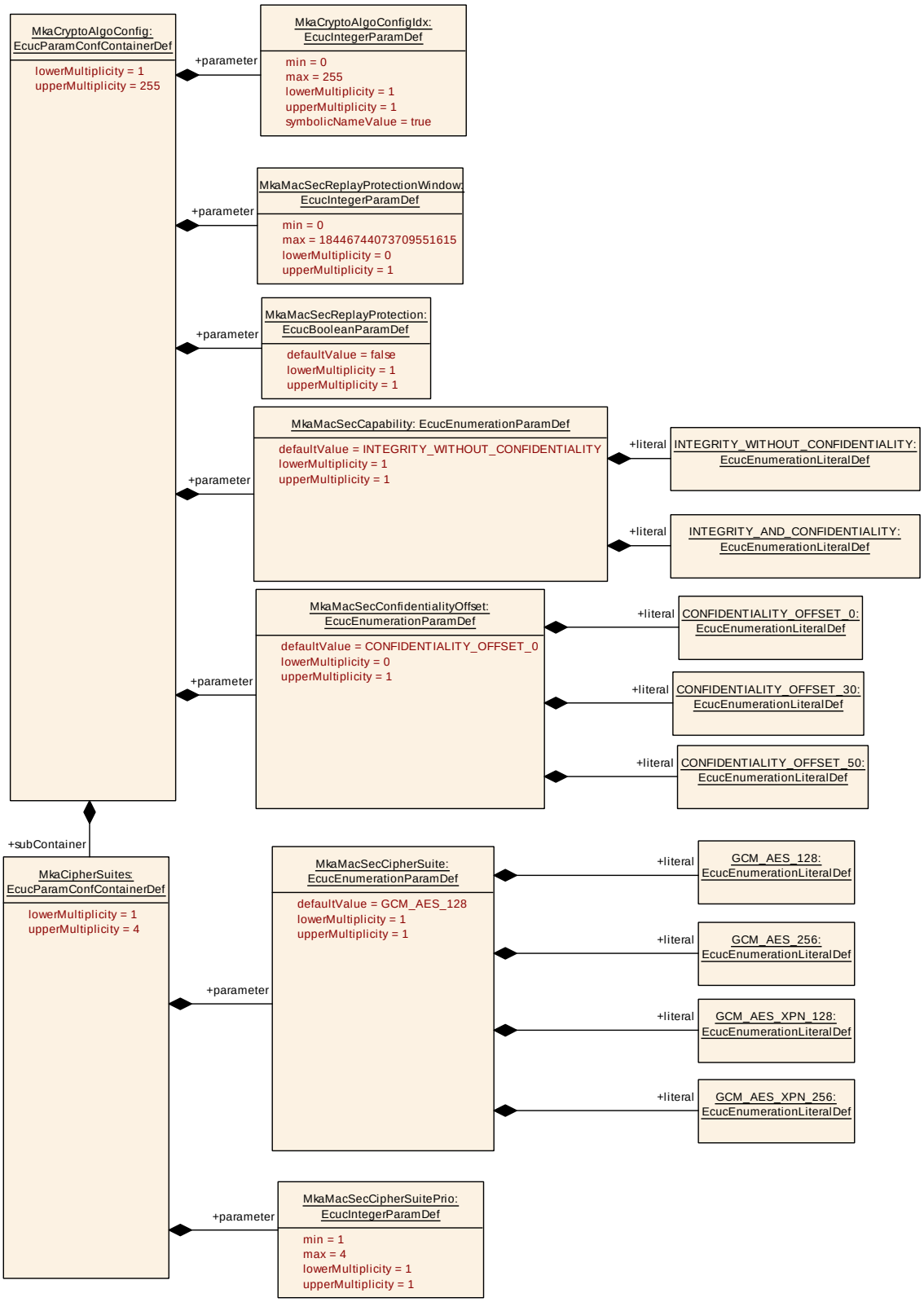

Figure 10.4: MkaCryptoAlgoConfig

10.2.5 MkaCipherSuites

[ECUC_Mka_00050] Definition of EcucParamConfContainerDef MkaCipherSuites

⌈

Container Name	MkaCipherSuites
Parent Container	MkaCryptoAlgoConfig
Upstream mapping defined	[8, CP_TPS_SystemTemplate]
Description	Cipher Suite configuration to use with MACsec. MkaCipherSuitePrio is present in case the MKA instance acts as a Key Server to select the cipher suite to use for MACsec.
Multiplicity	1..4
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
MkaMacSecCipherSuite	1	[ECUC_Mka_00052]
MkaMacSecCipherSuitePrio	1	[ECUC_Mka_00051]

No Included Containers

⌋

[ECUC_Mka_00052] Definition of EcucEnumerationParamDef MkaMacSecCipher Suite

Status: DRAFT

⌈

Parameter Name	MkaMacSecCipherSuite	
Parent Container	MkaCipherSuites	
Upstream mapping defined	[8, CP_TPS_SystemTemplate]	
Description	Cipher Suite to use for MACsec. **Tags:** atp.Status=draft	
Multiplicity	1	
Type	EcucEnumerationParamDef	
Range	GCM_AES_128	GCM AES 128 **Tags:** atp.Status=draft
	GCM_AES_256	GCM AES 256 **Tags:** atp.Status=draft
	GCM_AES_XPN_128	GCM AES XPN 128 (XPN = eXtended Packet Numbering, 64 Bit packet numbers) **Tags:** atp.Status=draft
	GCM_AES_XPN_256	GCM AES XPN 256 (XPN = eXtended Packet Numbering, 64 Bit packet numbers) **Tags:** atp.Status=draft
Default value	GCM_AES_128	
Pre-Compile only	true	

⌋

[ECUC_Mka_00051] Definition of EcucIntegerParamDef MkaMacSecCipherSuitePrio

Status: DRAFT

⌈

Parameter Name	MkaMacSecCipherSuitePrio
Parent Container	MkaCipherSuites
Upstream mapping defined	[8, CP_TPS_SystemTemplate]
Description	In case the MKA instance acts as a Key Server, the priority is used to select the Cipher Suite to use with MACsec from the common supported Ciphers (with the client in the link). Value of 1 means the highest priority. Value of 4 means the lowest priority. **Tags:** atp.Status=draft
Multiplicity	1
Type	EcucIntegerParamDef
Range	1 .. 4
Default value	–
Pre-Compile only	true

⌋

10.2.6 MkaPaeInstance

[ECUC_Mka_00003] Definition of EcucParamConfContainerDef MkaPaeInstance

Status: DRAFT

⌈

Container Name	MkaPaeInstance
Parent Container	Mka
Upstream mapping defined	[8, CP_TPS_SystemTemplate]
Description	MKA configuration of a controlled port. **Tags:** atp.Status=draft
Multiplicity	1..255
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
MkaOnFailPermissiveMode	1	[ECUC_Mka_00018]
MkaOnFailPermissiveModeTimeout	1	[ECUC_Mka_00019]
MkaPaeIdx	1	[ECUC_Mka_00011]
MkaEthIfControllerRef	0..1	[ECUC_Mka_00013]
MkaPaeConfRef	1	[ECUC_Mka_00054]
MkaSwitchPortRef	0..1	[ECUC_Mka_00014]

Included Containers		
Container Name	**Multiplicity**	**Description**
MkaKay	1	MKA instance (KaY) for a controlled port (PaE). **Tags:** atp.Status=draft
MkaPaeRxPdu	1	PDU used for reception of Ethernet frames. Supported Meta DataItemTypes: • ETHERNET_MAC_64 **Tags:** atp.Status=draft
MkaPaeTxPdu	1	PDU used for transmission of Ethernet frames. Supported Meta DataItemTypes: • ETHERNET_MAC_64 **Tags:** atp.Status=draft

⌋

[ECUC_Mka_00018] Definition of EcucEnumerationParamDef MkaOnFailPermissiveMode

Status: DRAFT

⌈

Parameter Name	MkaOnFailPermissiveMode	
Parent Container	MkaPaeInstance	
Upstream mapping defined	[8, CP_TPS_SystemTemplate]	
Description	Sets the behavior of the PAE in case MKA does not succeed when MKA is enabled. **Tags:** atp.Status=draft	
Multiplicity	1	
Type	EcucEnumerationParamDef	
Range	MKA_PERMISSIVE_MODE_NEVER	The controlled port will never be set to enabled if the participants cannot establish and successfully use a MACsec Secure Channel. **Tags:** atp.Status=draft
	MKA_PERMISSIVE_MODE_TIMEOUT	The controlled port will be set to enabled and MACsec will not be used in the referred port if the timeout value (MkaOnFailPermissiveMode Timeout) is reached and none MKA instance under the PAE instance could success the following conditions: - A participant belonging to the same CA was recognized and authenticated. - A secure channel could be established. - Both participants can transmit and receive MACsec protected traffic through the SC. **Tags:** atp.Status=draft
Pre-Compile only	true	

⌋

[ECUC_Mka_00019] Definition of EcucFloatParamDef MkaOnFailPermissive ModeTimeout

Status: DRAFT

⌈

Parameter Name	MkaOnFailPermissiveModeTimeout	
Parent Container	MkaPaeInstance	
Upstream mapping defined	[8, CP_TPS_SystemTemplate]	
Description	Timeout in seconds to enable the controlled port in case MkaOnFailPermissiveMode is set to Timeout. **Tags:** atp.Status=draft	
Multiplicity	1	
Type	EcucFloatParamDef	
Range	[0 .. INF]	
Default value	255	
Pre-Compile only	true	

⌋

[ECUC_Mka_00011] Definition of EcucIntegerParamDef MkaPaeIdx

Status: DRAFT

⌈

Parameter Name	MkaPaeIdx	
Parent Container	MkaPaeInstance	
Upstream mapping defined	–	
Description	Instance ID of the configured PAE. **Tags:** atp.Status=draft	
Multiplicity	1	
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)	
Range	0 .. 255	
Default value	–	
Pre-Compile only	true	

⌋

[ECUC_Mka_00013] Definition of EcucReferenceDef MkaEthIfControllerRef

Status: DRAFT

⌈

Parameter Name	MkaEthIfControllerRef
Parent Container	MkaPaeInstance
Upstream mapping defined	[8, CP_TPS_SystemTemplate]
Description	A reference to the EthIfController is used to identify the MACsec protected Ethernet hardware entity either as PHY or as an Ethernet controller. **Tags:** atp.Status=draft
Multiplicity	0..1
Type	Symbolic name reference to EthIfController

▽

△

Pre-Compile only	true

⌋

[ECUC_Mka_00054] Definition of EcucReferenceDef MkaPaeConfRef

Status: DRAFT

⌈

Parameter Name	MkaPaeConfRef
Parent Container	MkaPaeInstance
Upstream mapping defined	–
Description	Reference to the applicable PAE configuration. **Tags:** atp.Status=draft
Multiplicity	1
Type	Reference to MkaPaeConfiguration
Pre-Compile only	true

⌋

[ECUC_Mka_00014] Definition of EcucReferenceDef MkaSwitchPortRef

Status: DRAFT

⌈

Parameter Name	MkaSwitchPortRef
Parent Container	MkaPaeInstance
Upstream mapping defined	[8, CP_TPS_SystemTemplate]
Description	A reference to the EthSwtPort is used to identity the MACsec protected Ethernet hardware entity as Ethernet switch port. **Tags:** atp.Status=draft
Multiplicity	0..1
Type	Symbolic name reference to EthSwtPort
Pre-Compile only	true

⌋

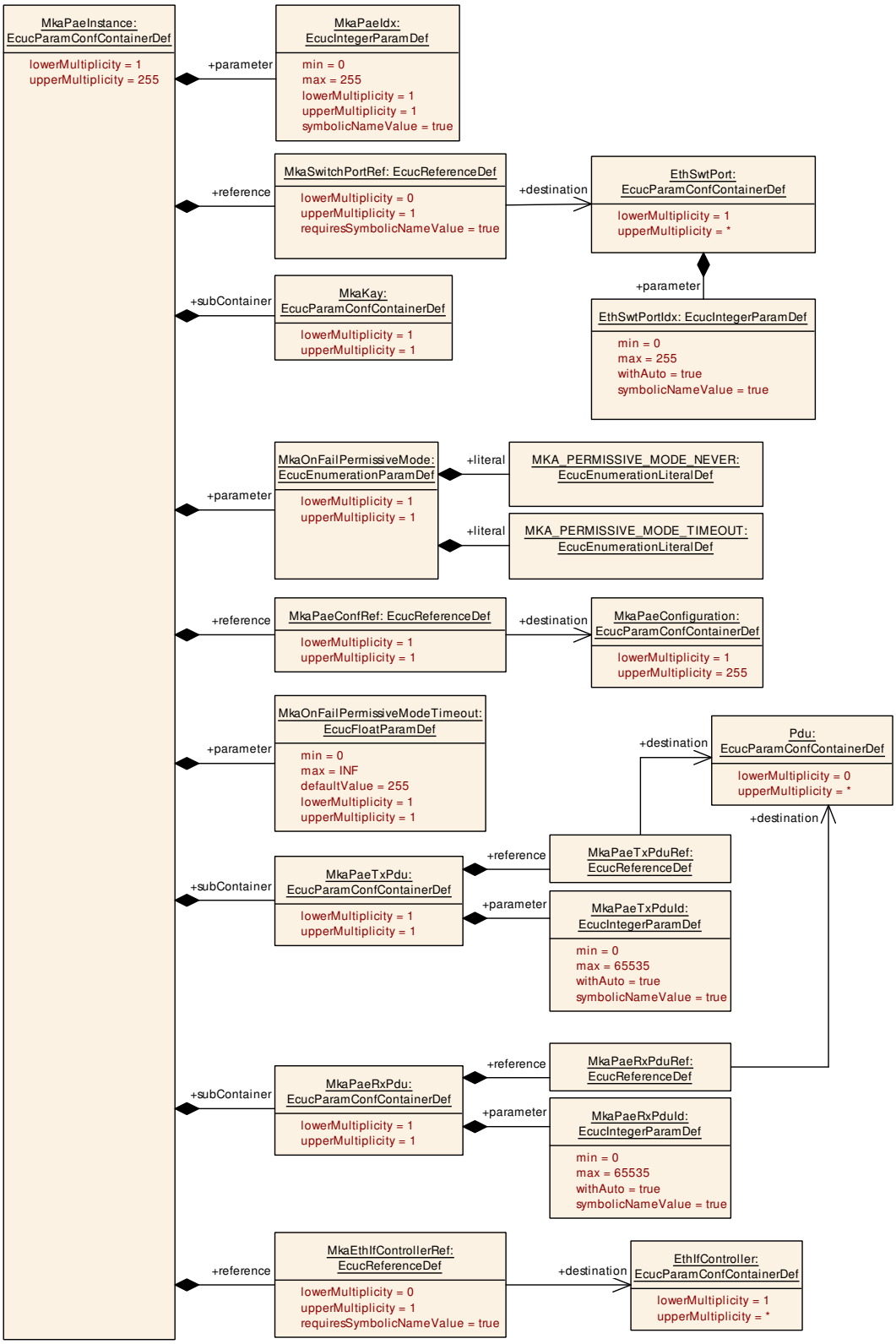

Figure 10.5: MkaPaeInstance

10.2.7 MkaPaeRxPdu

[ECUC_Mka_00071] Definition of EcucParamConfContainerDef MkaPaeRxPdu

Status: DRAFT

⌈

Container Name	MkaPaeRxPdu
Parent Container	MkaPaeInstance
Upstream mapping defined	–
Description	PDU used for reception of Ethernet frames. Supported MetaDataItemTypes: • ETHERNET_MAC_64 **Tags:** atp.Status=draft
Multiplicity	1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
MkaPaeRxPduId	1	[ECUC_Mka_00073]
MkaPaeRxPduRef	1	[ECUC_Mka_00072]

No Included Containers

⌋

[ECUC_Mka_00073] Definition of EcucIntegerParamDef MkaPaeRxPduId

Status: DRAFT

⌈

Parameter Name	MkaPaeRxPduId	
Parent Container	MkaPaeRxPdu	
Upstream mapping defined	–	
Description	The PDU identifier used for RxIndication from LSduR. **Tags:** atp.Status=draft	
Multiplicity	1	
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)	
Range	0 .. 65535	
Default value	–	
With Auto	true	
Pre-Compile only	true	

⌋

[ECUC_Mka_00072] Definition of EcucReferenceDef MkaPaeRxPduRef

Status: DRAFT

⌈

Parameter Name	MkaPaeRxPduRef
Parent Container	MkaPaeRxPdu
Upstream mapping defined	–
Description	Reference to the global PDU. **Tags:** atp.Status=draft
Multiplicity	1
Type	Reference to Pdu
Pre-Compile only	true

⌋

10.2.8 MkaPaeTxPdu

[ECUC_Mka_00068] Definition of EcucParamConfContainerDef MkaPaeTxPdu

Status: DRAFT

⌈

Container Name	MkaPaeTxPdu
Parent Container	MkaPaeInstance
Upstream mapping defined	–
Description	PDU used for transmission of Ethernet frames. Supported MetaDataItemTypes: • ETHERNET_MAC_64 **Tags:** atp.Status=draft
Multiplicity	1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
MkaPaeTxPduId	1	[ECUC_Mka_00070]
MkaPaeTxPduRef	1	[ECUC_Mka_00069]

No Included Containers

⌋

[ECUC_Mka_00070] Definition of EcucIntegerParamDef MkaPaeTxPduId

Status: DRAFT

⌈

Parameter Name	MkaPaeTxPduId
Parent Container	MkaPaeTxPdu
Upstream mapping defined	–
Description	The PDU identifier used for TxConfirmation from LSduR. **Tags:** atp.Status=draft
Multiplicity	1
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)
Range	0 .. 65535
Default value	–
With Auto	true
Pre-Compile only	true

⌋

[ECUC_Mka_00069] Definition of EcucReferenceDef MkaPaeTxPduRef

Status: DRAFT

⌈

Parameter Name	MkaPaeTxPduRef
Parent Container	MkaPaeTxPdu
Upstream mapping defined	–
Description	Reference to the global PDU. **Tags:** atp.Status=draft
Multiplicity	1
Type	Reference to Pdu
Pre-Compile only	true

⌋

10.2.9 MkaKay

[ECUC_Mka_00017] Definition of EcucParamConfContainerDef MkaKay

Status: DRAFT

⌈

Container Name	MkaKay
Parent Container	MkaPaeInstance
Upstream mapping defined	[8, CP_TPS_SystemTemplate]
Description	MKA instance (KaY) for a controlled port (PaE). **Tags:** atp.Status=draft
Multiplicity	1

▽

△

Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
MkaDstMacAddress	1	[ECUC_Mka_00032]
MkaKeyServerPriority	0..1	[ECUC_Mka_00022]
MkaRole	1	[ECUC_Mka_00029]

Included Containers		
Container Name	**Multiplicity**	**Description**
MkaKayDemEventParameterRefs	1	Container for the references to DemEventParameter elements which shall be invoked using the API Dem_SetEventStatus in case the corresponding error occurs. The EventId is taken from the referenced DemEventParameter's DemEventId symbolic value. The standardized errors are provided in this container and can be extended by vendor-specific error references. **Tags:** atp.Status=draft
MkaKayParticipant	1..255	MKA participant configuration. **Tags:** atp.Status=draft

⌋

[ECUC_Mka_00032] Definition of EcucStringParamDef MkaDstMacAddress

Status: DRAFT

⌈

Parameter Name	MkaDstMacAddress
Parent Container	MkaKay
Upstream mapping defined	[8, CP_TPS_SystemTemplate]
Description	Destination MAC address to use by the MKA instance. The destination MAC addresses to use are defined in the IEEE 802.1X-2020 chapter 11.1.1 (Table 11-1). **Tags:** atp.Status=draft
Multiplicity	1
Type	EcucStringParamDef
Default value	–
Regular Expression	–
Pre-Compile only	true

⌋

[ECUC_Mka_00022] Definition of EcucIntegerParamDef MkaKeyServerPriority

Status: DRAFT

⌈

Parameter Name	MkaKeyServerPriority	
Parent Container	MkaKay	
Upstream mapping defined	[8, CP_TPS_SystemTemplate]	
Description	Key Server Priority of the MKA participants. In case it is not provided, the default value is 0 for an MKA_KEY_SERVER and 255 for an MKA_PEER. **Tags:** atp.Status=draft	
Multiplicity	0..1	
Type	EcucIntegerParamDef	
Range	0 .. 255	
Default value	–	
Pre-Compile only	true	

⌋

[ECUC_Mka_00029] Definition of EcucEnumerationParamDef MkaRole

Status: DRAFT

⌈

Parameter Name	MkaRole	
Parent Container	MkaKay	
Upstream mapping defined	[8, CP_TPS_SystemTemplate]	
Description	Role of the MKA instance. **Tags:** atp.Status=draft	
Multiplicity	1	
Type	EcucEnumerationParamDef	
Range	MKA_KEY_SERVER	Act as an MKA key server **Tags:** atp.Status=draft
	MKA_PEER	Act as an MKA peer **Tags:** atp.Status=draft
Pre-Compile only	true	

⌋

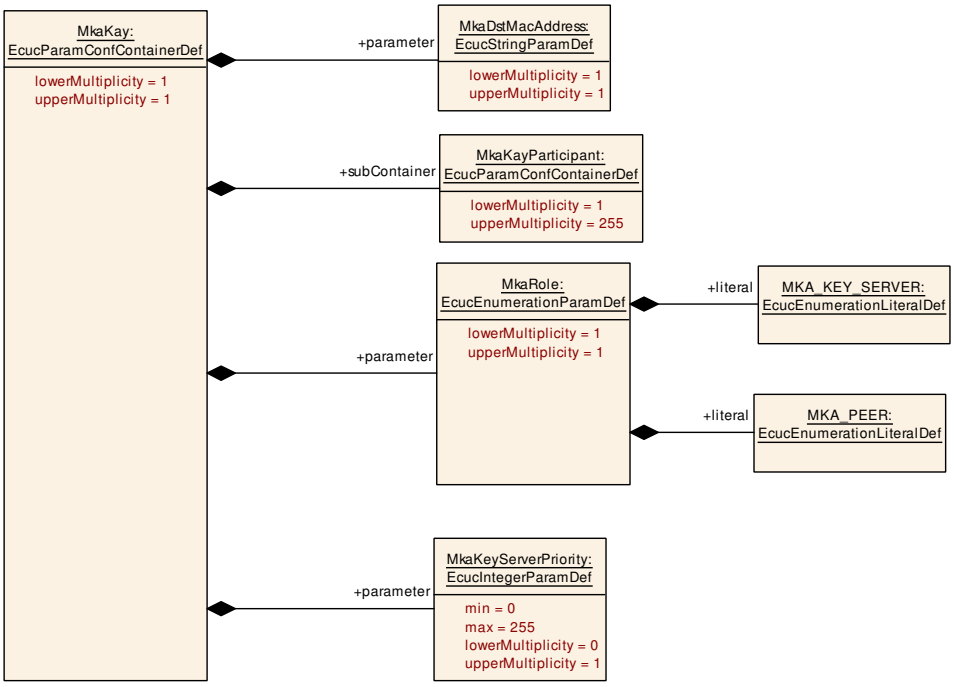

Figure 10.6: MkaKay

10.2.10 MkaKayParticipant

[ECUC_Mka_00038] Definition of EcucParamConfContainerDef MkaKayParticipant

Status: DRAFT

⌈

Container Name	MkaKayParticipant
Parent Container	MkaKay
Upstream mapping defined	[8, CP_TPS_SystemTemplate]
Description	MKA participant configuration. **Tags:** atp.Status=draft
Multiplicity	1..255
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
MkaParticipantActivate	1	[ECUC_Mka_00049]
MkaCryptoAlgoRef	1	[ECUC_Mka_00048]
MkaCryptoCknCakKeyRef	1	[ECUC_Mka_00040]
MkaCryptoHashKey128DerivationJobRef	0..1	[ECUC_Mka_00074]
MkaCryptoHashKey256DerivationJobRef	0..1	[ECUC_Mka_00075]

▽

△

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
MkaCryptoIckDeriveJobRef	1	[ECUC_Mka_00042]
MkaCryptoIcvGenerateJobRef	1	[ECUC_Mka_00043]
MkaCryptoIcvVerifyJobRef	1	[ECUC_Mka_00044]
MkaCryptoKekDeriveJobRef	1	[ECUC_Mka_00045]
MkaCryptoKeyUnwrapJobRef	0..1	[ECUC_Mka_00060]
MkaCryptoKeyWrapJobRef	0..1	[ECUC_Mka_00047]
MkaCryptoRandomJobRef	1	[ECUC_Mka_00041]
MkaCryptoSakKeyRef	1	[ECUC_Mka_00046]

No Included Containers

⌋

[ECUC_Mka_00049] Definition of EcucBooleanParamDef MkaParticipantActivate

Status: DRAFT

⌈

Parameter Name	MkaParticipantActivate
Parent Container	MkaKayParticipant
Upstream mapping defined	–
Description	Enabled/Disabled status of the MKA participant. - True = The MKA Participant exchanges MKPDUs - False = The MKA participant does not exchange MKPDUs. **Tags:** atp.Status=draft
Multiplicity	1
Type	EcucBooleanParamDef
Default value	false
Pre-Compile only	true

⌋

[ECUC_Mka_00048] Definition of EcucReferenceDef MkaCryptoAlgoRef

Status: DRAFT

⌈

Parameter Name	MkaCryptoAlgoRef
Parent Container	MkaKayParticipant
Upstream mapping defined	[8, CP_TPS_SystemTemplate]
Description	Reference to the cryptography to use (MkaAlgoConfiguration Container). **Tags:** atp.Status=draft
Multiplicity	1
Type	Reference to MkaCryptoAlgoConfig
Pre-Compile only	true

⌋

[ECUC_Mka_00040] Definition of EcucReferenceDef MkaCryptoCknCakKeyRef

Status: DRAFT

⌈

Parameter Name	MkaCryptoCknCakKeyRef
Parent Container	MkaKayParticipant
Upstream mapping defined	–
Description	Reference to the CKN (min. 1 & max. 32 characters) assigned to the KaY Participant in the CSM. **Tags:** atp.Status=draft
Multiplicity	1
Type	Symbolic name reference to CsmKey
Pre-Compile only	true

⌋

[ECUC_Mka_00074] Definition of EcucReferenceDef MkaCryptoHashKey128DerivationJobRef

Status: DRAFT

⌈

Parameter Name	MkaCryptoHashKey128DerivationJobRef
Parent Container	MkaKayParticipant
Upstream mapping defined	–
Description	Reference to a CsmEncrypt job for AES-128 cipher suites HashKey Derivation. **Tags:** atp.Status=draft
Multiplicity	0..1
Type	Symbolic name reference to CsmJob
Pre-Compile only	true

⌋

[ECUC_Mka_00075] Definition of EcucReferenceDef MkaCryptoHashKey256DerivationJobRef

Status: DRAFT

⌈

Parameter Name	MkaCryptoHashKey256DerivationJobRef
Parent Container	MkaKayParticipant
Upstream mapping defined	–
Description	Reference to a CsmEncrypt job for AES-256 cipher suites HashKey Derivation **Tags:** atp.Status=draft
Multiplicity	0..1
Type	Symbolic name reference to CsmJob
Pre-Compile only	true

⌋

[ECUC_Mka_00042] Definition of EcucReferenceDef MkaCryptoIckDeriveJobRef

Status: DRAFT

⌈

Parameter Name	MkaCryptoIckDeriveJobRef
Parent Container	MkaKayParticipant
Upstream mapping defined	–
Description	Reference to a CSM job for ICK Derivation. **Tags:** atp.Status=draft
Multiplicity	1
Type	Symbolic name reference to CsmJob
Pre-Compile only	true

⌋

[ECUC_Mka_00043] Definition of EcucReferenceDef MkaCryptoIcvGenerateJob Ref

Status: DRAFT

⌈

Parameter Name	MkaCryptoIcvGenerateJobRef
Parent Container	MkaKayParticipant
Upstream mapping defined	–
Description	Reference to a CSM job for ICV generation (according to IEEE_802.x ICV is always 128 bits). **Tags:** atp.Status=draft
Multiplicity	1
Type	Symbolic name reference to CsmJob
Pre-Compile only	true

⌋

[ECUC_Mka_00044] Definition of EcucReferenceDef MkaCryptoIcvVerifyJobRef

Status: DRAFT

⌈

Parameter Name	MkaCryptoIcvVerifyJobRef
Parent Container	MkaKayParticipant
Upstream mapping defined	–
Description	Reference to a CSM job for ICV verification (according to IEEE_802.x ICV is always 128 bits). **Tags:** atp.Status=draft
Multiplicity	1
Type	Symbolic name reference to CsmJob
Pre-Compile only	true

⌋

[ECUC_Mka_00045] Definition of EcucReferenceDef MkaCryptoKekDeriveJobRef

Status: DRAFT

⌈

Parameter Name	MkaCryptoKekDeriveJobRef
Parent Container	MkaKayParticipant
Upstream mapping defined	–
Description	Reference to a CSM job for KEK Derivation. (Note: CAK needs to be set as the KEK Derive job CsmJobKeyRef) **Tags:** atp.Status=draft
Multiplicity	1
Type	Symbolic name reference to CsmJob
Pre-Compile only	true

⌋

[ECUC_Mka_00060] Definition of EcucReferenceDef MkaCryptoKeyUnwrapJobRef

Status: DRAFT

⌈

Parameter Name	MkaCryptoKeyUnwrapJobRef
Parent Container	MkaKayParticipant
Upstream mapping defined	–
Description	Reference to a CSM job for SAK unwrap (to perform the Decrypt part of RFC3394). **Tags:** atp.Status=draft Dependency: MkaCryptoKeyUnwrapJobRef shall exist if MkaRole = MKA_PEER
Multiplicity	0..1
Type	Symbolic name reference to CsmJob
Pre-Compile only	true

⌋

[ECUC_Mka_00047] Definition of EcucReferenceDef MkaCryptoKeyWrapJobRef

Status: DRAFT

⌈

Parameter Name	MkaCryptoKeyWrapJobRef
Parent Container	MkaKayParticipant
Upstream mapping defined	–
Description	Reference to a CSM job for SAK wrap (to perform the Encrypt part of RFC3394). **Tags:** atp.Status=draft Dependency: MkaCryptoKeyWrapJobRef shall exist if MkaRole = MKA_KEY_SERVER
Multiplicity	0..1
Type	Symbolic name reference to CsmJob
Pre-Compile only	true

⌋

[ECUC_Mka_00041] Definition of EcucReferenceDef MkaCryptoRandomJobRef

Status: DRAFT

⌈

Parameter Name	MkaCryptoRandomJobRef
Parent Container	MkaKayParticipant
Upstream mapping defined	–
Description	Reference to a CSM job for random number generation. **Tags:** atp.Status=draft
Multiplicity	1
Type	Symbolic name reference to CsmJob
Pre-Compile only	true

⌋

[ECUC_Mka_00046] Definition of EcucReferenceDef MkaCryptoSakKeyRef

Status: DRAFT

⌈

Parameter Name	MkaCryptoSakKeyRef
Parent Container	MkaKayParticipant
Upstream mapping defined	–
Description	Reference to a CSM key where SAK shall be stored. **Tags:** atp.Status=draft
Multiplicity	1
Type	Symbolic name reference to CsmKey
Pre-Compile only	true

⌋

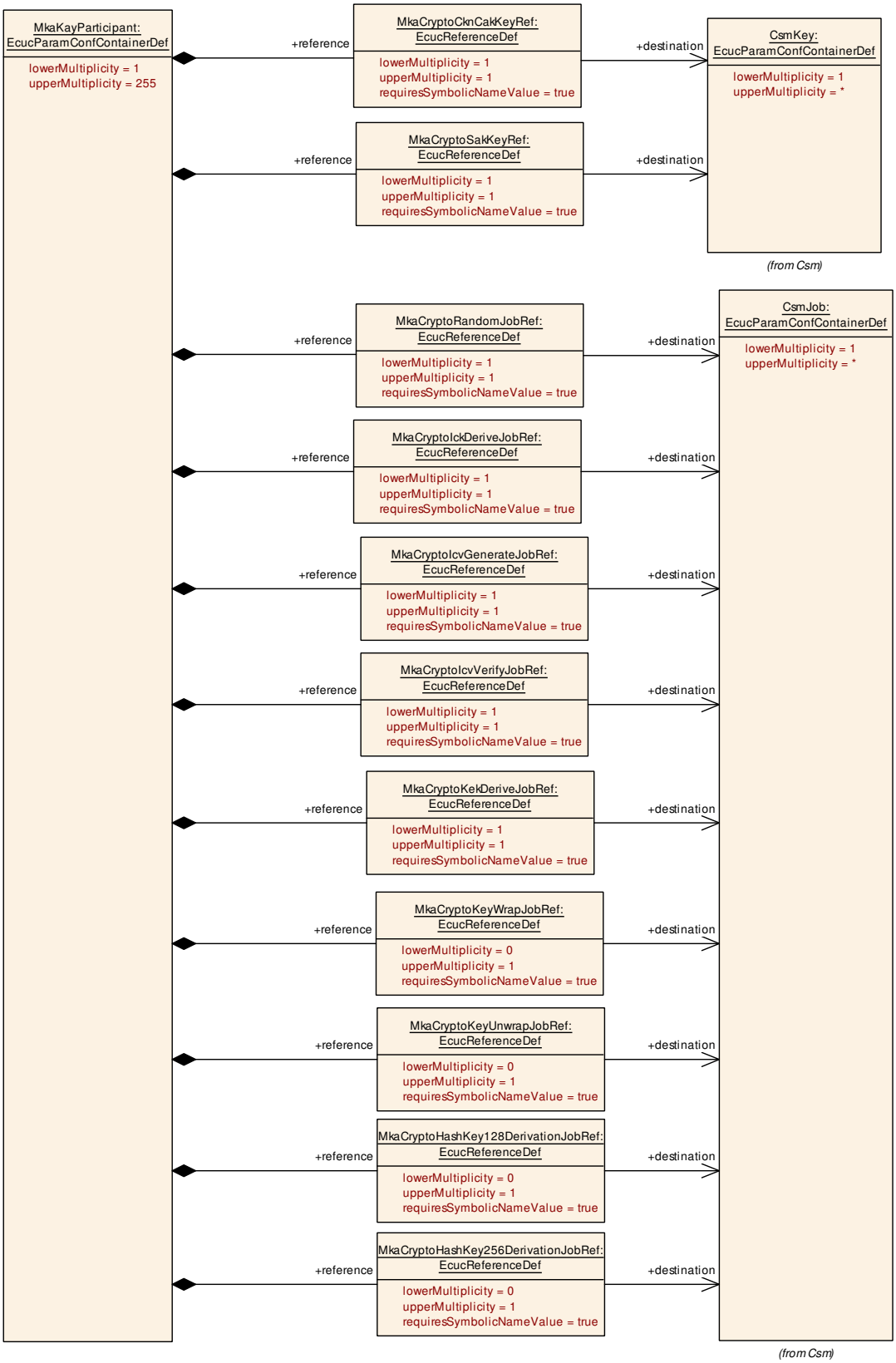

Figure 10.7: MkaKayParticipant

10.2.11 MkaKayDemEventParameterRefs

[ECUC_Mka_00055] Definition of EcucParamConfContainerDef MkaKayDemEventParameterRefs

Status: DRAFT

⌈

Container Name	MkaKayDemEventParameterRefs
Parent Container	MkaKay
Upstream mapping defined	–
Description	Container for the references to DemEventParameter elements which shall be invoked using the API Dem_SetEventStatus in case the corresponding error occurs. The Event Id is taken from the referenced DemEventParameter's DemEventId symbolic value. The standardized errors are provided in this container and can be extended by vendor-specific error references. **Tags:** atp.Status=draft
Multiplicity	1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
MKA_E_ALGO_MISMATCH_INSTANCE	0..1	[ECUC_Mka_00059]
MKA_E_KEY_MISMATCH_INSTANCE	0..1	[ECUC_Mka_00058]
MKA_E_KEY_NOT_PRESENT_INSTANCE	0..1	[ECUC_Mka_00057]
MKA_E_TIMEOUT_INSTANCE	0..1	[ECUC_Mka_00056]

No Included Containers

⌋

[ECUC_Mka_00059] Definition of EcucReferenceDef MKA_E_ALGO_MISMATCH_INSTANCE

Status: DRAFT

⌈

Parameter Name	MKA_E_ALGO_MISMATCH_INSTANCE
Parent Container	MkaKayDemEventParameterRefs
Upstream mapping defined	–
Description	Reference to the DemEventParameter which shall be issued when the MkaKay Instance does not successfully agree on MACsec keys and at least one MkaKay Participant does not support a common MACsec cipher suite. **Tags:** atp.Status=draft
Multiplicity	0..1
Type	Symbolic name reference to DemEventParameter
Pre-Compile only	true

⌋

[ECUC_Mka_00058] Definition of EcucReferenceDef MKA_E_KEY_MISMATCH_INSTANCE

Status: DRAFT

⌈

Parameter Name	MKA_E_KEY_MISMATCH_INSTANCE
Parent Container	MkaKayDemEventParameterRefs
Upstream mapping defined	–
Description	Reference to the DemEventParameter which shall be issued when the MkaKay Instance does not successfully agree on MACsec keys and at least one exchange for this MkaKay Instance encounters an ICV validation failure. **Tags:** atp.Status=draft
Multiplicity	0..1
Type	Symbolic name reference to DemEventParameter
Pre-Compile only	true

⌋

[ECUC_Mka_00057] Definition of EcucReferenceDef MKA_E_KEY_NOT_PRESENT_INSTANCE

Status: DRAFT

⌈

Parameter Name	MKA_E_KEY_NOT_PRESENT_INSTANCE
Parent Container	MkaKayDemEventParameterRefs
Upstream mapping defined	–
Description	Reference to the DemEventParameter which shall be issued when the MkaKay Instance does not successfully agree on MACsec keys and at least one of the keys (CAK) for this MkaKay Instance is not present. **Tags:** atp.Status=draft
Multiplicity	0..1
Type	Symbolic name reference to DemEventParameter
Pre-Compile only	true

⌋

[ECUC_Mka_00056] Definition of EcucReferenceDef MKA_E_TIMEOUT_INSTANCE

Status: DRAFT

⌈

Parameter Name	MKA_E_TIMEOUT_INSTANCE
Parent Container	MkaKayDemEventParameterRefs
Upstream mapping defined	–
Description	Reference to the DemEventParameter which shall be issued when the MkaKay Instance does not successfully agree on MACsec keys and at least one exchange for this MkaKay Instance encounters a timeout. **Tags:** atp.Status=draft

▽

△

Multiplicity	0..1
Type	Symbolic name reference to DemEventParameter
Pre-Compile only	true

⌋

10.2.12 MkaSecurityEventRefs

[ECUC_Mka_00062] Definition of EcucParamConfContainerDef MkaSecurityEventRefs

Status: DRAFT

⌈

Container Name	MkaSecurityEventRefs
Parent Container	MkaGeneral
Upstream mapping defined	–
Description	Container for the references to IdsMEvent elements representing the security events that the Mka module shall report to the IdsM in case the corresponding security related event occurs (and if MkaEnableSecurityEventReporting is set to "true"). The standardized security events in this container can be extended by vendor-specific security events. **Tags:** atp.Status=draft
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
SEV_MKA_AUTHENTICATION_FAILURE	0..1	[ECUC_Mka_00063]
SEV_MKA_CIPHER_SUITE_NOT_SUPPORTED	0..1	[ECUC_Mka_00066]
SEV_MKA_PORT_NOT_ENABLED	0..1	[ECUC_Mka_00065]
SEV_MKA_PORT_NUMBER_CHANGE	0..1	[ECUC_Mka_00067]
SEV_MKA_TIMEOUT	0..1	[ECUC_Mka_00064]

No Included Containers

⌋

[ECUC_Mka_00063] Definition of EcucReferenceDef SEV_MKA_AUTHENTICATION_FAILURE

Status: DRAFT

⌈

Parameter Name	SEV_MKA_AUTHENTICATION_FAILURE
Parent Container	MkaSecurityEventRefs
Upstream mapping defined	–
Description	Event triggered when the authentication during the MKA communication has failed (wrong CKN/CAK) **Tags:** atp.Status=draft
Multiplicity	0..1
Type	Symbolic name reference to IdsMEvent
Pre-Compile only	true

⌋

[ECUC_Mka_00066] Definition of EcucReferenceDef SEV_MKA_CIPHER_SUITE_NOT_SUPPORTED

Status: DRAFT

⌈

Parameter Name	SEV_MKA_CIPHER_SUITE_NOT_SUPPORTED
Parent Container	MkaSecurityEventRefs
Upstream mapping defined	–
Description	Event triggered when there is no Cipher Suite supported **Tags:** atp.Status=draft
Multiplicity	0..1
Type	Symbolic name reference to IdsMEvent
Pre-Compile only	true

⌋

[ECUC_Mka_00065] Definition of EcucReferenceDef SEV_MKA_PORT_NOT_ENABLED

Status: DRAFT

⌈

Parameter Name	SEV_MKA_PORT_NOT_ENABLED
Parent Container	MkaSecurityEventRefs
Upstream mapping defined	–
Description	Event triggered when the indicated port for the MKA communication is not enabled **Tags:** atp.Status=draft
Multiplicity	0..1
Type	Symbolic name reference to IdsMEvent
Pre-Compile only	true

⌋

[ECUC_Mka_00067] Definition of EcucReferenceDef SEV_MKA_PORT_NUMBER_CHANGE

Status: DRAFT

⌈

Parameter Name	SEV_MKA_PORT_NUMBER_CHANGE
Parent Container	MkaSecurityEventRefs
Upstream mapping defined	–
Description	Event triggered when during the MKA communication the port number has changed **Tags:** atp.Status=draft
Multiplicity	0..1
Type	Symbolic name reference to IdsMEvent
Pre-Compile only	true

⌋

[ECUC_Mka_00064] Definition of EcucReferenceDef SEV_MKA_TIMEOUT

Status: DRAFT

⌈

Parameter Name	SEV_MKA_TIMEOUT
Parent Container	MkaSecurityEventRefs
Upstream mapping defined	–
Description	Event triggered when the timeout for the MKA communication has expired **Tags:** atp.Status=draft
Multiplicity	0..1
Type	Symbolic name reference to IdsMEvent
Pre-Compile only	true

⌋

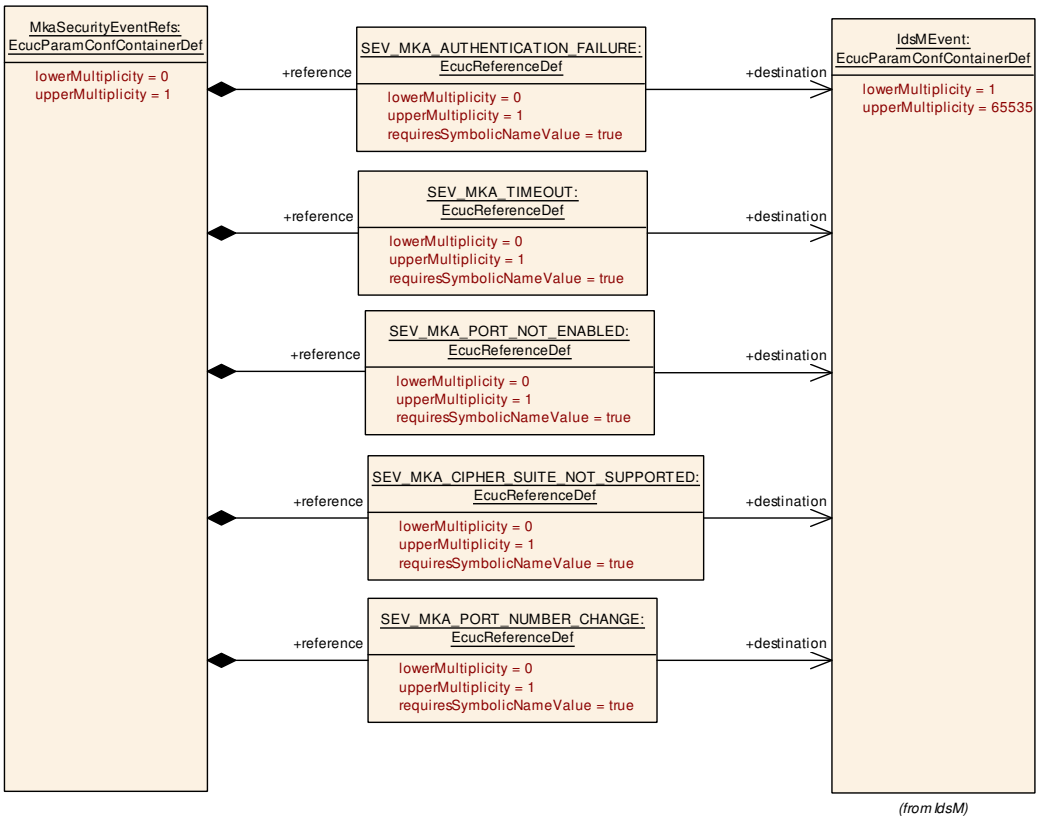

Figure 10.8: MkaSecurityEventRefs

10.3 Published Information

For details refer to [2] Chapter 10.3 *"Published Information"*.

A Not applicable requirements

[CP_SWS_Mka_00999]

Upstream requirements: FO_RS_MACsec_00001, FO_RS_MACsec_00011, FO_RS_MACsec_00012, FO_RS_MACsec_00018, FO_RS_MACsec_00019, FO_RS_MACsec_00021, FO_RS_MACsec_00022, FO_RS_MACsec_00034, FO_RS_MACsec_00035, FO_RS_MACsec_00036

⌈These requirements are not applicable to this specification.⌋

B Change history of AUTOSAR traceable items

Please note that the lists in this chapter also include traceable items that have been removed from the specification in a later version. These items do not appear as hyperlinks in the document.

B.1 Traceable item history of this document according to AUTOSAR Release R23-11

B.1.1 Added Specification Items in R23-11

Number	Heading
[CP_SWS_Mka_00301]	Security event report
[CP_SWS_Mka_00302]	Security events for Mka
[CP_SWS_Mka_00303]	Context data of security events for MKA
[CP_SWS_Mka_00304]	Description of security event SEV_MKA_AUTHENTICATION_FAILURE
[CP_SWS_Mka_00305]	Description of security event SEV_MKA_TIMEOUT
[CP_SWS_Mka_00306]	Description of security event SEV_MKA_PORT_NOT_ENABLED
[CP_SWS_Mka_00307]	Description of security event SEV_MKA_CIPHER_SUITE_NOT_SUPPORTED
[CP_SWS_Mka_00308]	Description of security event SEV_MKA_PORT_NUMBER_CHANGE

Table B.1: Added Specification Items in R23-11

B.1.2 Changed Specification Items in R23-11

Number	Heading
[CP_SWS_Mka_91001]	Definition of API function Mka_Init
[CP_SWS_Mka_91014]	Definition of API function Mka_GetVersionInfo
[CP_SWS_Mka_91015]	Definition of API function Mka_SetCknStatus
[CP_SWS_Mka_91016]	Definition of API function Mka_GetCknStatus



△

Number	Heading
[CP_SWS_Mka_91017]	Definition of API function Mka_GetEnable
[CP_SWS_Mka_91018]	Definition of API function Mka_GetPaeStatus
[CP_SWS_Mka_91019]	Definition of API function Mka_GetMacSecStatistics
[CP_SWS_Mka_91020]	Definition of API function Mka_SetEnable
[CP_SWS_Mka_91021]	Definition of API function Mka_SetPaePermissiveMode
[CP_SWS_Mka_91022]	Definition of API function Mka_StartPae
[CP_SWS_Mka_91023]	Definition of API function Mka_LinkStateChange

Table B.2: Changed Specification Items in R23-11

B.1.3 Deleted Specification Items in R23-11

none

B.1.4 Added Constraints in R23-11

none

B.1.5 Changed Constraints in R23-11

none

B.1.6 Deleted Constraints in R23-11

none

B.2 Traceable item history of this document according to AUTOSAR Release R24-11

B.2.1 Added Constraints in R24-11

Number	Heading
[CP_SWS_Mka_CONSTR_00019]	
[CP_SWS_Mka_CONSTR_00020]	

Table B.3: Added Constraints in R24-11

B.2.2 Changed Constraints in R24-11

none

B.2.3 Deleted Constraints in R24-11

none

B.2.4 Added Specification Items in R24-11

Number	Heading
[CP_SWS_Mka_00001]	
[CP_SWS_Mka_00002]	
[CP_SWS_Mka_00003]	
[CP_SWS_Mka_00004]	
[CP_SWS_Mka_00005]	
[CP_SWS_Mka_00006]	
[CP_SWS_Mka_00007]	

▽

△

Number	Heading
[CP_SWS_Mka_00008]	
[CP_SWS_Mka_00009]	
[CP_SWS_Mka_00011]	
[CP_SWS_Mka_00012]	
[CP_SWS_Mka_00013]	
[CP_SWS_Mka_00014]	
[CP_SWS_Mka_00015]	
[CP_SWS_Mka_00016]	
[CP_SWS_Mka_00017]	
[CP_SWS_Mka_00022]	
[CP_SWS_Mka_00023]	
[CP_SWS_Mka_00024]	
[CP_SWS_Mka_00025]	
[CP_SWS_Mka_00026]	
[CP_SWS_Mka_00027]	
[CP_SWS_Mka_00028]	
[CP_SWS_Mka_00029]	
[CP_SWS_Mka_00030]	
[CP_SWS_Mka_00031]	
[CP_SWS_Mka_00032]	
[CP_SWS_Mka_00033]	
[CP_SWS_Mka_00034]	

▽

△

Number	Heading
[CP_SWS_Mka_00035]	
[CP_SWS_Mka_00036]	
[CP_SWS_Mka_00037]	
[CP_SWS_Mka_00200]	
[CP_SWS_Mka_00201]	
[CP_SWS_Mka_00202]	
[CP_SWS_Mka_00203]	
[CP_SWS_Mka_00301]	Security event report
[CP_SWS_Mka_00302]	Security events for Mka
[CP_SWS_Mka_00304]	Description of security event SEV_MKA_AUTHENTICATION_FAILURE
[CP_SWS_Mka_00305]	Description of security event SEV_MKA_TIMEOUT
[CP_SWS_Mka_00306]	Description of security event SEV_MKA_PORT_NOT_ENABLED
[CP_SWS_Mka_00307]	Description of security event SEV_MKA_CIPHER_SUITE_NOT_SUPPORTED
[CP_SWS_Mka_00308]	Description of security event SEV_MKA_PORT_NUMBER_CHANGE
[CP_SWS_Mka_00309]	Security event context data definition: SEV_MKA_AUTHENTICATION_FAILURE
[CP_SWS_Mka_00310]	Security event context data definition: SEV_MKA_TIMEOUT
[CP_SWS_Mka_00311]	Security event context data definition: SEV_MKA_PORT_NOT_ENABLED
[CP_SWS_Mka_00312]	Security event context data definition: SEV_MKA_CIPHER_SUITE_NOT_SUPPORTED
[CP_SWS_Mka_00313]	Security event context data definition: SEV_MKA_PORT_NUMBER_CHANGE
[CP_SWS_Mka_00999]	
[CP_SWS_Mka_01001]	
[CP_SWS_Mka_01002]	

▽

△

Number	Heading
[CP_SWS_Mka_01003]	
[CP_SWS_Mka_01004]	
[CP_SWS_Mka_01005]	
[CP_SWS_Mka_01006]	
[CP_SWS_Mka_01007]	
[CP_SWS_Mka_01008]	
[CP_SWS_Mka_01009]	
[CP_SWS_Mka_01010]	
[CP_SWS_Mka_01011]	
[CP_SWS_Mka_01012]	
[CP_SWS_Mka_01013]	
[CP_SWS_Mka_91001]	Definition of API function Mka_Init
[CP_SWS_Mka_91002]	Definition of datatype Mka_MacSecConfigType
[CP_SWS_Mka_91003]	Definition of datatype Mka_ConfidentialityOffsetType
[CP_SWS_Mka_91004]	Definition of datatype Mka_ValidateFramesType
[CP_SWS_Mka_91005]	Definition of imported datatypes of module Mka
[CP_SWS_Mka_91006]	Definition of mandatory interfaces required by module Mka
[CP_SWS_Mka_91007]	Definition of optional interfaces requested by module Mka
[CP_SWS_Mka_91008]	Definition of datatype Mka_Stats_Tx_SecYType
[CP_SWS_Mka_91009]	Definition of datatype Mka_Stats_Tx_ScType
[CP_SWS_Mka_91010]	Definition of datatype Mka_Stats_Rx_SecYType
[CP_SWS_Mka_91011]	Definition of datatype Mka_Stats_Rx_ScType

▽

△

Number	Heading
[CP_SWS_Mka_91012]	Definition of datatype Mka_PermissiveModeType
[CP_SWS_Mka_91013]	Definition of datatype Mka_SakKeyPtrType
[CP_SWS_Mka_91014]	Definition of API function Mka_GetVersionInfo
[CP_SWS_Mka_91015]	Definition of API function Mka_SetCknStatus
[CP_SWS_Mka_91016]	Definition of API function Mka_GetCknStatus
[CP_SWS_Mka_91017]	Definition of API function Mka_GetEnable
[CP_SWS_Mka_91018]	Definition of API function Mka_GetPaeStatus
[CP_SWS_Mka_91019]	Definition of API function Mka_GetMacSecStatistics
[CP_SWS_Mka_91020]	Definition of API function Mka_SetEnable
[CP_SWS_Mka_91021]	Definition of API function Mka_SetPaePermissiveMode
[CP_SWS_Mka_91022]	Definition of API function Mka_StartPae
[CP_SWS_Mka_91023]	Definition of API function Mka_LinkStateChange
[CP_SWS_Mka_91024]	Definition of callback function Mka_GetMacSecStatisticsNotification
[CP_SWS_Mka_91025]	Definition of datatype Mka_MkaStatus
[CP_SWS_Mka_91026]	Definition of datatype Mka_ConfigType
[CP_SWS_Mka_91027]	Definition of datatype Mka_PaeStatusType
[CP_SWS_Mka_91028]	Definition of datatype Mka_Stats_SecYType
[CP_SWS_Mka_91029]	Definition of callback function Mka_RxIndication
[CP_SWS_Mka_91030]	Definition of callback function Mka_TxConfirmation
[CP_SWS_Mka_91031]	Definition of callback function Mka_MacSecUpdateSecYNotification
[CP_SWS_Mka_91032]	Definition of callback function Mka_MacSecAddTxSaNotification
[CP_SWS_Mka_91033]	Definition of callback function Mka_MacSecAddRxSaNotification

▽

△

Number	Heading
[CP_SWS_Mka_91034]	Definition of scheduled function Mka_MainFunction
[CP_SWS_Mka_91035]	Definiton of development errors in module Mka
[CP_SWS_Mka_91036]	Definition of configurable interface <User_GetMacSecStatisticsCallback Notification>
[ECUC_Mka_00001]	Definition of EcucModuleDef Mka
[ECUC_Mka_00002]	Definition of EcucParamConfContainerDef MkaGeneral
[ECUC_Mka_00003]	Definition of EcucParamConfContainerDef MkaPaeInstance
[ECUC_Mka_00004]	Definition of EcucFloatParamDef MkaRetryBaseDelay
[ECUC_Mka_00005]	Definition of EcucFloatParamDef MkaRetryCyclicDelay
[ECUC_Mka_00007]	Definition of EcucFloatParamDef MkaHelloTime
[ECUC_Mka_00009]	Definition of EcucFloatParamDef MkaLifeTime
[ECUC_Mka_00010]	Definition of EcucFloatParamDef MkaSakRetireTime
[ECUC_Mka_00011]	Definition of EcucIntegerParamDef MkaPaeIdx
[ECUC_Mka_00012]	Definition of EcucBooleanParamDef MkaAutoStart
[ECUC_Mka_00013]	Definition of EcucReferenceDef MkaEthIfControllerRef
[ECUC_Mka_00014]	Definition of EcucReferenceDef MkaSwitchPortRef
[ECUC_Mka_00015]	Definition of EcucIntegerParamDef MkaBypassVlan
[ECUC_Mka_00016]	Definition of EcucIntegerParamDef MkaBypassEtherType
[ECUC_Mka_00017]	Definition of EcucParamConfContainerDef MkaKay
[ECUC_Mka_00018]	Definition of EcucEnumerationParamDef MkaOnFailPermissiveMode
[ECUC_Mka_00019]	Definition of EcucFloatParamDef MkaOnFailPermissiveModeTimeout
[ECUC_Mka_00021]	Definition of EcucParamConfContainerDef MkaCryptoAlgoConfig
[ECUC_Mka_00022]	Definition of EcucIntegerParamDef MkaKeyServerPriority
[ECUC_Mka_00024]	Definition of EcucFloatParamDef MkaSakRekeyTimeSpan
[ECUC_Mka_00025]	Definition of EcucEnumerationParamDef MkaMacSecCapability
[ECUC_Mka_00026]	Definition of EcucEnumerationParamDef MkaMacSecConfidentialityOffset
[ECUC_Mka_00027]	Definition of EcucBooleanParamDef MkaMacSecReplayProtection
[ECUC_Mka_00028]	Definition of EcucIntegerParamDef MkaMacSecReplayProtectionWindow
[ECUC_Mka_00029]	Definition of EcucEnumerationParamDef MkaRole
[ECUC_Mka_00031]	Definition of EcucStringParamDef MkaSrcMacAddress
[ECUC_Mka_00032]	Definition of EcucStringParamDef MkaDstMacAddress
[ECUC_Mka_00033]	Definition of EcucParamConfContainerDef MkaPaeConfiguration
[ECUC_Mka_00034]	Definition of EcucBooleanParamDef MkaDevErrorDetect
[ECUC_Mka_00035]	Definition of EcucFloatParamDef MkaMainFunctionPeriod
[ECUC_Mka_00036]	Definition of EcucBooleanParamDef MkaVersionInfoApi
[ECUC_Mka_00037]	Definition of EcucIntegerParamDef MkaPaeConfigurationIdx
[ECUC_Mka_00038]	Definition of EcucParamConfContainerDef MkaKayParticipant

▽

△

Number	Heading
[ECUC_Mka_00040]	Definition of EcucReferenceDef MkaCryptoCknCakKeyRef
[ECUC_Mka_00041]	Definition of EcucReferenceDef MkaCryptoRandomJobRef
[ECUC_Mka_00042]	Definition of EcucReferenceDef MkaCryptoIckDeriveJobRef
[ECUC_Mka_00043]	Definition of EcucReferenceDef MkaCryptoIcvGenerateJobRef
[ECUC_Mka_00044]	Definition of EcucReferenceDef MkaCryptoIcvVerifyJobRef
[ECUC_Mka_00045]	Definition of EcucReferenceDef MkaCryptoKekDeriveJobRef
[ECUC_Mka_00046]	Definition of EcucReferenceDef MkaCryptoSakKeyRef
[ECUC_Mka_00047]	Definition of EcucReferenceDef MkaCryptoKeyWrapJobRef
[ECUC_Mka_00048]	Definition of EcucReferenceDef MkaCryptoAlgoRef
[ECUC_Mka_00049]	Definition of EcucBooleanParamDef MkaParticipantActivate
[ECUC_Mka_00050]	Definition of EcucParamConfContainerDef MkaCipherSuites
[ECUC_Mka_00051]	Definition of EcucIntegerParamDef MkaMacSecCipherSuitePrio
[ECUC_Mka_00052]	Definition of EcucEnumerationParamDef MkaMacSecCipherSuite
[ECUC_Mka_00053]	Definition of EcucIntegerParamDef MkaCryptoAlgoConfigIdx
[ECUC_Mka_00054]	Definition of EcucReferenceDef MkaPaeConfRef
[ECUC_Mka_00055]	Definition of EcucParamConfContainerDef MkaKayDemEventParameterRefs
[ECUC_Mka_00056]	Definition of EcucReferenceDef MKA_E_TIMEOUT_INSTANCE
[ECUC_Mka_00057]	Definition of EcucReferenceDef MKA_E_KEY_NOT_PRESENT_INSTANCE
[ECUC_Mka_00058]	Definition of EcucReferenceDef MKA_E_KEY_MISMATCH_INSTANCE
[ECUC_Mka_00059]	Definition of EcucReferenceDef MKA_E_ALGO_MISMATCH_INSTANCE
[ECUC_Mka_00060]	Definition of EcucReferenceDef MkaCryptoKeyUnwrapJobRef
[ECUC_Mka_00061]	Definition of EcucBooleanParamDef MkaEnableSecurityEventReporting
[ECUC_Mka_00062]	Definition of EcucParamConfContainerDef MkaSecurityEventRefs
[ECUC_Mka_00063]	Definition of EcucReferenceDef SEV_MKA_AUTHENTICATION_FAILURE
[ECUC_Mka_00064]	Definition of EcucReferenceDef SEV_MKA_TIMEOUT
[ECUC_Mka_00065]	Definition of EcucReferenceDef SEV_MKA_PORT_NOT_ENABLED
[ECUC_Mka_00066]	Definition of EcucReferenceDef SEV_MKA_CIPHER_SUITE_NOT_SUPPORTED
[ECUC_Mka_00067]	Definition of EcucReferenceDef SEV_MKA_PORT_NUMBER_CHANGE
[ECUC_Mka_00068]	Definition of EcucParamConfContainerDef MkaPaeTxPdu
[ECUC_Mka_00069]	Definition of EcucReferenceDef MkaPaeTxPduRef
[ECUC_Mka_00070]	Definition of EcucIntegerParamDef MkaPaeTxPduId
[ECUC_Mka_00071]	Definition of EcucParamConfContainerDef MkaPaeRxPdu
[ECUC_Mka_00072]	Definition of EcucReferenceDef MkaPaeRxPduRef
[ECUC_Mka_00073]	Definition of EcucIntegerParamDef MkaPaeRxPduId
[ECUC_Mka_00074]	Definition of EcucReferenceDef MkaCryptoHashKey128DerivationJobRef
[ECUC_Mka_00075]	Definition of EcucReferenceDef MkaCryptoHashKey256DerivationJobRef

▽

△

Number	Heading
[ECUC_Mka_00076]	Definition of EcucFunctionNameDef MkaGetMacSecStatisticsCallback Notification

Table B.4: Added Specification Items in R24-11

B.2.5 Changed Specification Items in R24-11

none

B.2.6 Deleted Specification Items in R24-11

none

B.3 Traceable item history of this document according to AUTOSAR Release R25-11

B.3.1 Added Constraints in R25-11

Number	Heading
[CP_SWS_Mka_CONSTR_00021]	`MkaPaeRxPdu` and `MkaPaeTxPdu` shall refer to an `EthIfController` that allows MACsec unprotected traffic
[CP_SWS_Mka_CONSTR_00022]	`MkaPaeInstance` shall either reference an `EthIfController` or an `EthSwtPort`

Table B.5: Added Constraints in R25-11

B.3.2 Changed Constraints in R25-11

none

B.3.3 Deleted Constraints in R25-11

none

B.3.4 Added Specification Items in R25-11

Number	Heading
[CP_SWS_Mka_00038]	MAC source / destination address of a MKPDU to be transmitted
[CP_SWS_Mka_00039]	Call of `LSduR_MkaTransmit` for a generated MKPDU
[CP_SWS_Mka_00040]	Retry to trigger transmission for a call of `LSduR_MkaTransmit`
[CP_SWS_Mka_00041]	Retry of transmission for a generated MKPDU where the corresponding `Mka_TxConfirmation` returns with `result` set to `E_NOT_OK`
[CP_SWS_Mka_00042]	Update the MAC address filter in the Ethernet controller with the IEEE specified MAC multicast address for MKPDU exchange
[CP_SWS_Mka_00043]	Retry to update the MAC address filter if `EthIf_UpdatePhysAddrFilter` returned with `E_NOT_OK`
[CP_SWS_Mka_00044]	Retry of function calls which return with `E_NOT_OK`
[CP_SWS_Mka_00045]	Considerations to provide a SAK key reference of type `Mka_SakKeyPtrType`
[CP_SWS_Mka_00047]	Shutdown MACsec operational functionality if `MkaPaeInstance` references an `EthIfController`
[CP_SWS_Mka_00048]	Shutdown MACsec operational functionality if `MkaPaeInstance` references an `EthSwtPort`
[CP_SWS_Mka_00049]	Trigger to reset the configuration of a MACsec protected port if link state changes to DOWN
[ECUC_Mka_00077]	Definition of EcucBooleanParamDef MkaCalculateSakHashKey

Table B.6: Added Specification Items in R25-11

B.3.5 Changed Specification Items in R25-11

Number	Heading
[CP_SWS_Mka_00031]	Trigger to configure a MACsec protected port if link state changes to ACTIVE
[CP_SWS_Mka_00032]	Signal a successful finalization of a configuration for a MACsec protected port
[CP_SWS_Mka_00999]	
[CP_SWS_Mka_91002]	Definition of datatype Mka_MacSecConfigType
[CP_SWS_Mka_91005]	Definition of imported datatypes of module Mka
[CP_SWS_Mka_91006]	Definition of mandatory interfaces required by module Mka

▽

△

Number	Heading
[CP_SWS_Mka_91007]	Definition of optional interfaces requested by module Mka
[CP_SWS_Mka_91013]	Definition of datatype Mka_SakKeyPtrType
[CP_SWS_Mka_91024]	Definition of callback function Mka_GetMacSecStatisticsNotification
[CP_SWS_Mka_91031]	Definition of callback function Mka_MacSecUpdateSecYNotification
[CP_SWS_Mka_91032]	Definition of callback function Mka_MacSecAddTxSaNotification
[CP_SWS_Mka_91033]	Definition of callback function Mka_MacSecAddRxSaNotification
[ECUC_Mka_00002]	Definition of EcucParamConfContainerDef MkaGeneral
[ECUC_Mka_00003]	Definition of EcucParamConfContainerDef MkaPaeInstance
[ECUC_Mka_00013]	Definition of EcucReferenceDef MkaEthIfControllerRef
[ECUC_Mka_00014]	Definition of EcucReferenceDef MkaSwitchPortRef
[ECUC_Mka_00017]	Definition of EcucParamConfContainerDef MkaKay
[ECUC_Mka_00021]	Definition of EcucParamConfContainerDef MkaCryptoAlgoConfig
[ECUC_Mka_00033]	Definition of EcucParamConfContainerDef MkaPaeConfiguration
[ECUC_Mka_00038]	Definition of EcucParamConfContainerDef MkaKayParticipant
[ECUC_Mka_00047]	Definition of EcucReferenceDef MkaCryptoKeyWrapJobRef
[ECUC_Mka_00050]	Definition of EcucParamConfContainerDef MkaCipherSuites
[ECUC_Mka_00055]	Definition of EcucParamConfContainerDef MkaKayDemEventParameterRefs
[ECUC_Mka_00060]	Definition of EcucReferenceDef MkaCryptoKeyUnwrapJobRef
[ECUC_Mka_00062]	Definition of EcucParamConfContainerDef MkaSecurityEventRefs
[ECUC_Mka_00068]	Definition of EcucParamConfContainerDef MkaPaeTxPdu
[ECUC_Mka_00071]	Definition of EcucParamConfContainerDef MkaPaeRxPdu

Table B.7: Changed Specification Items in R25-11

B.3.6 Deleted Specification Items in R25-11

Number	Heading
[CP_SWS_Mka_00003]	
[CP_SWS_Mka_00004]	
[CP_SWS_Mka_00023]	
[ECUC_Mka_00015]	Definition of EcucIntegerParamDef MkaBypassVlan

▽

△

Number	Heading
[ECUC_Mka_00016]	Definition of EcucIntegerParamDef MkaBypassEtherType
[ECUC_Mka_00031]	Definition of EcucStringParamDef MkaSrcMacAddress

Table B.8: Deleted Specification Items in R25-11

B.4 Traceable item history of this document according to AUTOSAR Release R26-11

B.4.1 Added Constraints in R26-11

Number	Heading
[CP_SWS_Mka_CONSTR_00023]	Provide MAC destination address of a MKPDU via meta data at transmission side
[CP_SWS_Mka_CONSTR_00024]	Consume MAC source address of a MKPDU via meta data at reception side

Table B.9: Added Constraints in R26-11

B.4.2 Changed Constraints in R26-11

none

B.4.3 Deleted Constraints in R26-11

none

B.4.4 Added Specification Items in R26-11

Number	Heading
[CP_SWS_Mka_00050]	Consideration for ICV calculation at transmission side
[CP_SWS_Mka_00051]	Consideration for ICV validation at reception side

▽

△

Number	Heading
[SWS_Mka_91044]	Definition of callback function Mka_CanSecUpdateSecYNotification
[SWS_Mka_91045]	Definition of callback function Mka_CanSecAddTxSaNotification
[SWS_Mka_91046]	Definition of callback function Mka_CanSecAddRxSaNotification
[SWS_Mka_91047]	Definition of callback function Mka_GetCanSecSecYStatisticsNotification
[SWS_Mka_91048]	Definition of callback function Mka_GetCanSecTxScStatisticsNotification
[SWS_Mka_91049]	Definition of callback function Mka_GetCanSecRxScStatisticsNotification
[SWS_Mka_91050]	Definition of callback function Mka_CanControllerModeIndication
[SWS_Mka_91051]	Definition of datatype Mka_CanSecConfigType

Table B.10: Added Specification Items in R26-11

B.4.5 Changed Specification Items in R26-11

Number	Heading
[CP_SWS_Mka_00042]	Update the MAC address filter in the Ethernet controller with the configured MAC multicast address for MKPDU exchange
[CP_SWS_Mka_00301]	Security event report
[CP_SWS_Mka_00302]	Security events for Mka
[CP_SWS_Mka_00304]	Description of security event SEV_MKA_AUTHENTICATION_FAILURE
[CP_SWS_Mka_00305]	Description of security event SEV_MKA_TIMEOUT
[CP_SWS_Mka_00306]	Description of security event SEV_MKA_PORT_NOT_ENABLED
[CP_SWS_Mka_00307]	Description of security event SEV_MKA_CIPHER_SUITE_NOT_SUPPORTED
[CP_SWS_Mka_00308]	Description of security event SEV_MKA_PORT_NUMBER_CHANGE
[CP_SWS_Mka_00309]	Security event context data definition: SEV_MKA_AUTHENTICATION_FAILURE
[CP_SWS_Mka_00310]	Security event context data definition: SEV_MKA_TIMEOUT
[CP_SWS_Mka_00311]	Security event context data definition: SEV_MKA_PORT_NOT_ENABLED
[CP_SWS_Mka_00312]	Security event context data definition: SEV_MKA_CIPHER_SUITE_NOT_SUPPORTED
[CP_SWS_Mka_00313]	Security event context data definition: SEV_MKA_PORT_NUMBER_CHANGE

▽

△

Number	Heading
[CP_SWS_Mka_91005]	Definition of imported datatypes of module Mka
[CP_SWS_Mka_91007]	Definition of optional interfaces requested by module Mka
[CP_SWS_Mka_91030]	Definition of callback function Mka_TxConfirmation
[ECUC_Mka_00001]	Definition of EcucModuleDef Mka
[ECUC_Mka_00002]	Definition of EcucParamConfContainerDef MkaGeneral
[ECUC_Mka_00003]	Definition of EcucParamConfContainerDef MkaPaeInstance
[ECUC_Mka_00004]	Definition of EcucFloatParamDef MkaRetryBaseDelay
[ECUC_Mka_00005]	Definition of EcucFloatParamDef MkaRetryCyclicDelay
[ECUC_Mka_00007]	Definition of EcucFloatParamDef MkaHelloTime
[ECUC_Mka_00009]	Definition of EcucFloatParamDef MkaLifeTime
[ECUC_Mka_00010]	Definition of EcucFloatParamDef MkaSakRetireTime
[ECUC_Mka_00011]	Definition of EcucIntegerParamDef MkaPaeIdx
[ECUC_Mka_00012]	Definition of EcucBooleanParamDef MkaAutoStart
[ECUC_Mka_00013]	Definition of EcucReferenceDef MkaEthIfControllerRef
[ECUC_Mka_00014]	Definition of EcucReferenceDef MkaSwitchPortRef
[ECUC_Mka_00017]	Definition of EcucParamConfContainerDef MkaKay
[ECUC_Mka_00018]	Definition of EcucEnumerationParamDef MkaOnFailPermissiveMode
[ECUC_Mka_00019]	Definition of EcucFloatParamDef MkaOnFailPermissiveModeTimeout
[ECUC_Mka_00021]	Definition of EcucParamConfContainerDef MkaCryptoAlgoConfig
[ECUC_Mka_00022]	Definition of EcucIntegerParamDef MkaKeyServerPriority
[ECUC_Mka_00024]	Definition of EcucFloatParamDef MkaSakRekeyTimeSpan
[ECUC_Mka_00025]	Definition of EcucEnumerationParamDef MkaMacSecCapability
[ECUC_Mka_00026]	Definition of EcucEnumerationParamDef MkaMacSecConfidentialityOffset
[ECUC_Mka_00027]	Definition of EcucBooleanParamDef MkaMacSecReplayProtection
[ECUC_Mka_00028]	Definition of EcucIntegerParamDef MkaMacSecReplayProtectionWindow
[ECUC_Mka_00029]	Definition of EcucEnumerationParamDef MkaRole
[ECUC_Mka_00032]	Definition of EcucStringParamDef MkaDstMacAddress
[ECUC_Mka_00033]	Definition of EcucParamConfContainerDef MkaPaeConfiguration
[ECUC_Mka_00034]	Definition of EcucBooleanParamDef MkaDevErrorDetect
[ECUC_Mka_00035]	Definition of EcucFloatParamDef MkaMainFunctionPeriod
[ECUC_Mka_00037]	Definition of EcucIntegerParamDef MkaPaeConfigurationIdx
[ECUC_Mka_00038]	Definition of EcucParamConfContainerDef MkaKayParticipant
[ECUC_Mka_00040]	Definition of EcucReferenceDef MkaCryptoCknCakKeyRef
[ECUC_Mka_00041]	Definition of EcucReferenceDef MkaCryptoRandomJobRef
[ECUC_Mka_00042]	Definition of EcucReferenceDef MkaCryptoIckDeriveJobRef
[ECUC_Mka_00043]	Definition of EcucReferenceDef MkaCryptoIcvGenerateJobRef

▽

△

Number	Heading
[ECUC_Mka_00044]	Definition of EcucReferenceDef MkaCryptoIcvVerifyJobRef
[ECUC_Mka_00045]	Definition of EcucReferenceDef MkaCryptoKekDeriveJobRef
[ECUC_Mka_00046]	Definition of EcucReferenceDef MkaCryptoSakKeyRef
[ECUC_Mka_00047]	Definition of EcucReferenceDef MkaCryptoKeyWrapJobRef
[ECUC_Mka_00048]	Definition of EcucReferenceDef MkaCryptoAlgoRef
[ECUC_Mka_00049]	Definition of EcucBooleanParamDef MkaParticipantActivate
[ECUC_Mka_00050]	Definition of EcucParamConfContainerDef MkaCipherSuites
[ECUC_Mka_00051]	Definition of EcucIntegerParamDef MkaMacSecCipherSuitePrio
[ECUC_Mka_00052]	Definition of EcucEnumerationParamDef MkaMacSecCipherSuite
[ECUC_Mka_00053]	Definition of EcucIntegerParamDef MkaCryptoAlgoConfigIdx
[ECUC_Mka_00054]	Definition of EcucReferenceDef MkaPaeConfRef
[ECUC_Mka_00055]	Definition of EcucParamConfContainerDef MkaKayDemEventParameterRefs
[ECUC_Mka_00056]	Definition of EcucReferenceDef MKA_E_TIMEOUT_INSTANCE
[ECUC_Mka_00057]	Definition of EcucReferenceDef MKA_E_KEY_NOT_PRESENT_INSTANCE
[ECUC_Mka_00058]	Definition of EcucReferenceDef MKA_E_KEY_MISMATCH_INSTANCE
[ECUC_Mka_00059]	Definition of EcucReferenceDef MKA_E_ALGO_MISMATCH_INSTANCE
[ECUC_Mka_00060]	Definition of EcucReferenceDef MkaCryptoKeyUnwrapJobRef
[ECUC_Mka_00061]	Definition of EcucBooleanParamDef MkaEnableSecurityEventReporting
[ECUC_Mka_00062]	Definition of EcucParamConfContainerDef MkaSecurityEventRefs
[ECUC_Mka_00063]	Definition of EcucReferenceDef SEV_MKA_AUTHENTICATION_FAILURE
[ECUC_Mka_00064]	Definition of EcucReferenceDef SEV_MKA_TIMEOUT
[ECUC_Mka_00065]	Definition of EcucReferenceDef SEV_MKA_PORT_NOT_ENABLED
[ECUC_Mka_00066]	Definition of EcucReferenceDef SEV_MKA_CIPHER_SUITE_NOT_SUPPORTED
[ECUC_Mka_00067]	Definition of EcucReferenceDef SEV_MKA_PORT_NUMBER_CHANGE
[ECUC_Mka_00068]	Definition of EcucParamConfContainerDef MkaPaeTxPdu
[ECUC_Mka_00069]	Definition of EcucReferenceDef MkaPaeTxPduRef
[ECUC_Mka_00070]	Definition of EcucIntegerParamDef MkaPaeTxPduId
[ECUC_Mka_00071]	Definition of EcucParamConfContainerDef MkaPaeRxPdu
[ECUC_Mka_00072]	Definition of EcucReferenceDef MkaPaeRxPduRef
[ECUC_Mka_00073]	Definition of EcucIntegerParamDef MkaPaeRxPduId
[ECUC_Mka_00074]	Definition of EcucReferenceDef MkaCryptoHashKey128DerivationJobRef
[ECUC_Mka_00075]	Definition of EcucReferenceDef MkaCryptoHashKey256DerivationJobRef
[ECUC_Mka_00076]	Definition of EcucFunctionNameDef MkaGetMacSecStatisticsCallback Notification
[ECUC_Mka_00077]	Definition of EcucBooleanParamDef MkaCalculateSakHashKey

Table B.11: Changed Specification Items in R26-11

B.4.6 Deleted Specification Items in R26-11

Number	Heading
[CP_SWS_Mka_91014]	Definition of API function Mka_GetVersionInfo
[ECUC_Mka_00036]	Definition of EcucBooleanParamDef MkaVersionInfoApi

Table B.12: Deleted Specification Items in R26-11