Document Title	Specification of Secure Onboard Communication
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	654

Document Status	published
Part of AUTOSAR Standard	Classic Platform
Part of Standard Release	R24-11

Document Change History

Date	Release	Changed by	Description
2024-11-27	R24-11	AUTOSAR Release Management	• Minor corrections / clarifications / editorial changes; For details please refer to the Change Documentation
2023-11-23	R23-11	AUTOSAR Release Management	• Updated naming of Security Events for IdsM • Minor corrections / clarifications / editorial changes; For details please refer to the Change Documentation
2022-11-24	R22-11	AUTOSAR Release Management	• Minor corrections / clarifications / editorial changes; For details please refer to the Change Documentation
2021-11-25	R21-11	AUTOSAR Release Management	• Minor corrections / clarifications / editorial changes; For details please refer to the Change Documentation
2020-11-30	R20-11	AUTOSAR Release Management	• Added Security Events for IdsM • Added additional freshness value use case • Added separate Mainfunction container for multi core • Minor corrections / clarifications / editorial changes; For details please refer to the Change Documentation • Changed Document Status from Final to published

▽

△

2019-11-28	R19-11	AUTOSAR Release Management	• Added option to send default authentication information • Added an authentic PDU length header • Added new options to override the verification status • Minor corrections / clarifications / editorial changes; For details please refer to the Change Documentation • Changed Document Status from Final to published
2018-10-31	4.4.0	AUTOSAR Release Management	• Handle Dynamic length PDUs • Added option to send wrong Authentication Information • Provide failed verification status to application. • Minor corrections / clarifications / editorial changes; For details please refer to the Change Documentation.
2017-12-08	4.3.1	AUTOSAR Release Management	• Clarify new authentication data layout with optional parameters. • Clarified the details for SW-C Freshness Value Manager (Section 11). • Minor corrections / clarifications / editorial changes; For details please refer to the Change Documentation.
2016-11-30	4.3.0	AUTOSAR Release Management	• Handle freshness in external freshness manager • New feature to send authenticator in an additional message • Secured diagnostic communication • Increase minimum value of parameter AuthInfoTxLength to 1 • Changed the type of the parameter keyID of the interface SecOC_AssociateKey() to uint16

▽

△

2015-07-31	4.2.2	AUTOSAR Release Management	• Minor corrections / clarifications / editorial changes; For details please refer to the Change Documentation
2014-10-31	4.2.1	AUTOSAR Release Management	• Initial Release

Disclaimer

Contents

1 Introduction and functional overview

This specification is the AUTOSAR Secure Onboard Communication (SecOC) module Software Specification. It is based on AUTOSAR SecOC [1] and specifies how the requirements of the AUTOSAR SecOC SRS shall be realized. It describes the basic security features, the functionality and the API of the AUTOSAR SecOC module.

The SecOC module aims for resource-efficient and practicable authentication mechanisms for critical data on the level of PDUs. The authentication mechanisms shall be seamlessly integrated with the current AUTOSAR communication systems. The impact with respect to resource consumption should be as small as possible in order to allow protection as add-on for legacy systems. The specification is based on the assumption that mainly symmetric authentication approaches with message authentication codes (MACs) are used. They achieve the same level of security with much smaller keys than asymmetric approaches and can be implemented compactly and efficiently in software and in hardware. However, the specification provides the necessary level of abstraction so that both, symmetric approaches as well as asymmetric authentication approaches can be used.

The SecOC module integrates on the level of the AUTOSAR PduR. Figure 1.1 shows the integration of the SecOC module as part of the Autosar communication stack.

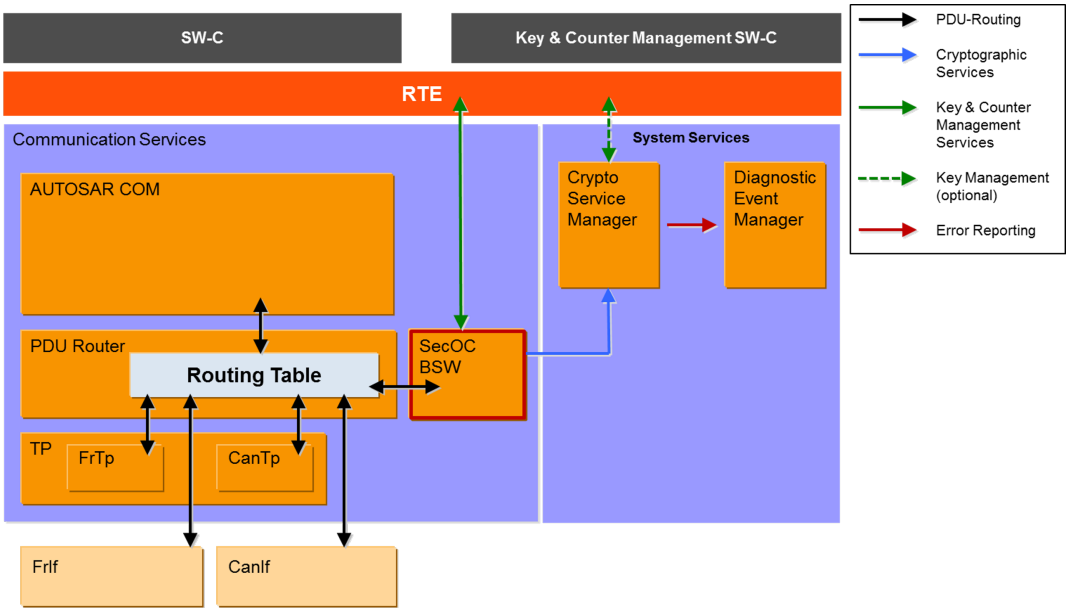

Figure 1.1: Integration of the SecOC BSW

In this setting,PduR is responsible to route incoming and outgoing security related I-PDUs to the SecOC module. The SecOC module shall then add or process the security relevant information and shall propagate the results in the form of an I-PDU back to the PduR. PduR is then responsible to further route the I-PDUs.Moreover, the SecOC module makes use of the cryptographic services provided by the CSM and interacts with the Rte to allow key and counter management. The SecOC module shall support all kind of communication paradigms and principles that are supported by PduR, es-

pecially Multicast communications, Transport Protocols and the PduR Gateway. The following sections provide a detailed specification of SecOC interfaces, functionality and configuration.

2 Acronyms, abbreviations and definitions

2.1 Acronyms and abbreviations

The glossary below includes acronyms and abbreviations relevant to the SecOC module that are not included in the [2, AUTOSAR glossary].

Abbreviation / Acronym:	Description:
CSM	The AUTOSAR Crypto Service Manager
SecOC	Secure Onboard Communication
MAC	Message Authentication Code
FV	Freshness Value
FM	Freshness Manager

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

2.2 Definitions

For this document the definitions of data integrity, authentication, entity authentication, data origin, message authentication and transaction authentication from [3] are used:

Term:	Description:
Authentic I-PDU	An Authentic I-PDU is an arbitrary AUTOSAR I-PDU the content of which is secured during network transmission by means of the Secured I-PDU. The secured content comprises the complete I-PDU or a part of the I-PDU.
Authentication	Authentication is a service related to identification. This function applies to both entities and information itself. Two parties entering into a communication should identify each other. Information delivered over a channel should be authenticated as to origin, date of origin, data content, time sent, etc. For these reasons, this aspect of cryptography is usually subdivided into two major classes: entity authentication and data origin authentication. Data origin authentication implicitly provides data integrity (for if a message is modified, the source has changed).
Authentication Information	The Authentication Information consists of a Freshness Value (or a part thereof) and an Authenticator (or a part thereof). Authentication Information are the additional pieces of information that are added by SecOC to realize the Secured I-PDU
Authenticator	Authenticator is data that is used to provide message authentication. In general, the term Message Authentication Code (MAC) is used for symmetric approaches while the term Signature or Digital Signature refers to asymmetric approaches having different properties and constraints.
Data integrity	Data integrity is the property whereby data has not been altered in an unauthorized manner since the time it was created, transmitted, or stored by an authorized source. To assure data integrity, one should have the ability to detect data manipulation by unauthorized parties. Data manipulation includes such things as insertion, deletion, and substitution.
Data origin authentication	Data origin authentication is a type of authentication whereby a party is corroborated as the (original) source of specified data created at some (typically unspecified) time in the past. By definition, data origin authentication includes data integrity.

△

Term:	Description:
Distinction unilateral/ bilateral authentication	In unilateral authentication, one side proves identity. The requesting side is not even authenticated to the extent of proving that it is allowed to request authentication. In bilateral authentication, the requester is also authenticated at least (see below) to prove the privilege of requesting. There is an efficient and more secure way to authenticate both endpoints, based on the bilateral authentication described above. Along with the authentication (in the second message) requested initially by the receiver (in the first message), the sender also requests an authentication. The receiver sends a third message providing the authentication requested by the sender. This is only three messages (in contrast to four with two unilateral messages).
Entity authentication	Entity authentication is the process whereby one party is assured (through acquisition of corroborative evidence) of the identity of a second party involved in a protocol, and that the second has actually participated (i.e., is active at, or immediately prior to, the time the evidence is acquired). Note: Entity authentication means to prove presence and operational readiness of a communication endpoint. This is for example often done by proving access to a cryptographic key and knowledge of a secret.It is necessary to do this without disclosing either key or secret.Entity authentication can be used to prevent record-and-replay attacks. Freshness of messages only complicates them by the need to record a lifetime and corrupt either senders or receivers (real-time) clock.Entity authentication is triggered by the receiver, i.e. the one to be convinced, while the sender has to react by convincing. Record and replay attacks on entity authentication are usually prevented by allowing the receiver some control over the authentication process.In order to prevent the receiver from using this control for steering the sender to malicious purposes or from determining a key or a secret ("oracle attack"), the sender can add more randomness.If not only access to a key (implying membership to a privileged group) but also individuality is to be proven, the sender additionally adds and authenticates its unique identification.
Message authentication	Message authentication is a term used analogously with data origin authentication. It provides data origin authentication with respect to the original message source (and data integrity, but no uniqueness and timeliness guarantees).
Secured I-PDU	A Secured I-PDU is an AUTOSAR I-PDU that contains Payload of an Authentic I-PDU supplemented by additional Authentication Information.
Transaction authentication	Transaction authentication denotes message authentication augmented to additionally provide uniqueness and timeliness guarantees on data (thus preventing undetectable message replay).

Table 2.2: Definitions used in the scope of this Document

3 Related documentation

3.1 Input documents & related standards and norms

[1] Requirements on Secure Onboard Communication
AUTOSAR_CP_RS_SecureOnboardCommunication

[2] Glossary
AUTOSAR_FO_TR_Glossary

[3] Handbook of Applied Cryptography
http://www.cacr.math.uwaterloo.ca/hac/

[4] General Specification of Basic Software Modules
AUTOSAR_CP_SWS_BSWGeneral

[5] General Requirements on Basic Software Modules
AUTOSAR_CP_RS_BSWGeneral

[6] NIST Special Publication 800-38B: Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication
http://csrc.nist.gov/publications/nistpubs/800-38B/SP_800-38B.pdf

[7] NIST: Announcing the Advanced Encryption Standard (AES)
http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf

3.2 Related specification

AUTOSAR provides a General Specification on Basic Software modules [4, SWS BSW General], which is also valid for the SecOC module.

Thus, the specification SWS BSW General shall be considered as additional and required specification for the SecOC module.

4 Constraints and assumptions

This document is applicable for AUTOSAR release 4.3.

4.1 Applicability to car domains

The SecOC module is used in all ECUs where secure communication is necessary.

The SecOC module has not been specified to work with MOST and LIN communication networks. With MOST not being specifically supported, the applicability to multimedia and telematic car domains may be limited.

4.2 SomeIpTp constraints

The SecOC module can only be used to secure the whole SomeIpTp message and cannot be used to secure individual segments of a SomeIpTp message.

Following module sequence on transmission side is allowed:

SecOC -> PduR -> SomeIpTp

Following module sequence on transmission side is not allowed:

SomeIpTp -> PduR -> SecOC

The main reason why the SecOC cannot be used to secure SomeIpTp individual message segments is the following one:

- The SomeIpTp requires a call of `SomeIpTp_TriggerTransmit` to create the SomIpTp header. The SecOC does not support the data provision via TriggerTransmit from the upper layer.

5 Dependencies to other modules

This chapter lists all the features from other modules that are used by the AUTOSAR SecOC module and functionalities that are provided by the AUTOSAR SecOC module to other modules. Because the SecOC module deals with I-PDUs that are either sourced or sunk by other modules, care should be taken that shared configuration items are consistent between the modules.

5.1 Dependencies to PduR

The SecOC module depends on the API and capabilities of the PduR. It provides the upper and lower layer API functions required by the PDU Router, namely

- the API of the communication interface modules,
- the API of the Transport Protocol Modules,
- the API of the upper layer modules which use transport protocol modules,
- the API of the upper layer modules which process I-PDUs originating from communication interface modules.

To serve the PduR with the results of the security processing, the SecOC module requires the respective API function of the PduR.

5.2 Dependencies to CSM

The SecOC module depends on cryptographic algorithms that are provided in AUTOSAR by the CSM module. The SecOC module requires API functions to generate and verify Cryptographic Signatures or Message Authentication Codes, namely

- the MAC-generate interface (`Csm_MacGenerate`),
- the MAC-verify interface (`Csm_MacVerify`),
- the Signature-generate interface (`Csm_SignatureGenerate`),
- the Signature-verify interface (`Csm_SignatureVerify`),

5.3 Dependencies to the RTE

The SecOC module provides an API with management functions. This API contains the following API functions that are provided as Service Interfaces by the RTE.

- `VerificationStatus`

- SecOC_VerifyStatusOverride.
- VerificationStatusIndication

The API functions are specified in more detail in Section 8.

The Rte includes the BSW-Scheduler. The SecOC module relies on the BSW-scheduler calling the functions SecOC_MainFunctionRx and SecOC_MainFunctionTx at a period as configured in SecOCMainFunctionPeriodRx and SecOCMainFunctionPeriodTx.

6 Requirements Tracing

The following tables reference the requirements specified in [5] and [1] 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
[RS_Ids_00810]	Basic SW security events	[SWS_SecOC_00115] [SWS_SecOC_00273] [SWS_SecOC_92000]
[SRS_BSW_00003]	All software modules shall provide version and identification information	[SWS_SecOC_00107]
[SRS_BSW_00101]	The Basic Software Module shall be able to initialize variables and hardware in a separate initialization function	[SWS_SecOC_00106] [SWS_SecOC_00269]
[SRS_BSW_00171]	Optional functionality of a Basic-SW component that is not required in the ECU shall be configurable at pre-compile-time	[SWS_SecOC_00153]
[SRS_BSW_00301]	All AUTOSAR Basic Software Modules shall only import the necessary information	[SWS_SecOC_00103]
[SRS_BSW_00323]	All AUTOSAR Basic Software Modules shall check passed API parameters for validity	[SWS_SecOC_00106] [SWS_SecOC_00107] [SWS_SecOC_00112] [SWS_SecOC_00113] [SWS_SecOC_00122] [SWS_SecOC_00124] [SWS_SecOC_00125] [SWS_SecOC_00126] [SWS_SecOC_00127] [SWS_SecOC_00128] [SWS_SecOC_00129] [SWS_SecOC_00130] [SWS_SecOC_00152] [SWS_SecOC_00157] [SWS_SecOC_00161] [SWS_SecOC_91008] [SWS_SecOC_91009]
[SRS_BSW_00337]	Classification of development errors	[SWS_SecOC_00101] [SWS_SecOC_00114]
[SRS_BSW_00357]	For success/failure of an API call a standard return type shall be defined	[SWS_SecOC_00112] [SWS_SecOC_00113] [SWS_SecOC_00122] [SWS_SecOC_00127] [SWS_SecOC_00128] [SWS_SecOC_00129] [SWS_SecOC_00130] [SWS_SecOC_91008] [SWS_SecOC_91009]
[SRS_BSW_00358]	The return type of init() functions implemented by AUTOSAR Basic Software Modules shall be void	[SWS_SecOC_00106]
[SRS_BSW_00359]	Callback Function Return Types for AUTOSAR BSW	[SWS_SecOC_00106] [SWS_SecOC_00107] [SWS_SecOC_00119] [SWS_SecOC_00124] [SWS_SecOC_00125] [SWS_SecOC_00126] [SWS_SecOC_00152] [SWS_SecOC_00161]
[SRS_BSW_00369]	All AUTOSAR Basic Software Modules shall not return specific development error codes via the API	[SWS_SecOC_00107] [SWS_SecOC_00112] [SWS_SecOC_91008]
[SRS_BSW_00373]	The main processing function of each AUTOSAR Basic Software Module shall be named according the defined convention	[SWS_SecOC_00171] [SWS_SecOC_00176]
[SRS_BSW_00384]	The Basic Software Module specifications shall specify at least in the description which other modules they require	[SWS_SecOC_00137] [SWS_SecOC_00138]

△

Requirement	Description	Satisfied by
[SRS_BSW_00385]	List possible error notifications	[SWS_SecOC_00077] [SWS_SecOC_00089] [SWS_SecOC_00101] [SWS_SecOC_00108] [SWS_SecOC_00109] [SWS_SecOC_00114] [SWS_SecOC_00121] [SWS_SecOC_00151] [SWS_SecOC_00155] [SWS_SecOC_00181] [SWS_SecOC_00213] [SWS_SecOC_00256] [SWS_SecOC_00260] [SWS_SecOC_00263] [SWS_SecOC_00264] [SWS_SecOC_00265] [SWS_SecOC_00266]
[SRS_BSW_00386]	The BSW shall specify the configuration and conditions for detecting an error	[SWS_SecOC_00101] [SWS_SecOC_00114]
[SRS_BSW_00402]	Each module shall provide version information	[SWS_SecOC_00107]
[SRS_BSW_00407]	Each BSW module shall provide a function to read out the version information of a dedicated module implementation	[SWS_SecOC_00107]
[SRS_BSW_00414]	Init functions shall have a pointer to a configuration structure as single parameter	[SWS_SecOC_00106]
[SRS_BSW_00425]	The BSW module description template shall provide means to model the defined trigger conditions of schedulable objects	[SWS_SecOC_00171] [SWS_SecOC_00176]
[SRS_BSW_00426]	BSW Modules shall ensure data consistency of data which is shared between BSW modules	[SWS_SecOC_00110]
[SRS_BSW_00432]	Modules should have separate main processing functions for read/receive and write/transmit data path	[SWS_SecOC_00274] [SWS_SecOC_00276]
[SRS_BSW_00449]	BSW Service APIs used by Autosar Application Software shall return a Std_ReturnType	[SWS_SecOC_00112] [SWS_SecOC_00113] [SWS_SecOC_00122] [SWS_SecOC_00125] [SWS_SecOC_00127] [SWS_SecOC_00152] [SWS_SecOC_91008] [SWS_SecOC_91009]
[SRS_BSW_00457]	Callback functions of Application software components shall be invoked by the Basis SW	[SWS_SecOC_00012]
[SRS_SecOC_00001]	Selection of Authentic I-PDU	[SWS_SecOC_00104]
[SRS_SecOC_00002]	Range of verification retry by the receiver	[SWS_SecOC_00047] [SWS_SecOC_00094] [SWS_SecOC_00232] [SWS_SecOC_00233] [SWS_SecOC_91005]

▽

△

Requirement	Description	Satisfied by
[SRS_SecOC_00003]	Configuration of different security properties / requirements	[SWS_SecOC_00012] [SWS_SecOC_00104] [SWS_SecOC_00190] [SWS_SecOC_00191] [SWS_SecOC_00192] [SWS_SecOC_00193] [SWS_SecOC_00194] [SWS_SecOC_00230] [SWS_SecOC_00231] [SWS_SecOC_00232] [SWS_SecOC_00244] [SWS_SecOC_00245] [SWS_SecOC_00246] [SWS_SecOC_00247] [SWS_SecOC_00249] [SWS_SecOC_00250] [SWS_SecOC_00311] [SWS_SecOC_00312] [SWS_SecOC_00313] [SWS_SecOC_00314] [SWS_SecOC_91001] [SWS_SecOC_91002] [SWS_SecOC_91003] [SWS_SecOC_91004] [SWS_SecOC_91005] [SWS_SecOC_91006] [SWS_SecOC_91007] [SWS_SecOC_91010] [SWS_SecOC_91012] [SWS_SecOC_91014] [SWS_SecOC_91015] [SWS_SecOC_91016] [SWS_SecOC_91020] [SWS_SecOC_91021] [SWS_SecOC_91022]
[SRS_SecOC_00005]	Initialisation of security information	[SWS_SecOC_00054] [SWS_SecOC_00162] [SWS_SecOC_00172] [SWS_SecOC_00177] [SWS_SecOC_00226] [SWS_SecOC_00235]
[SRS_SecOC_00006]	Creation of a Secured I-PDU from an Authentic I-PDU	[SWS_SecOC_00011] [SWS_SecOC_00031] [SWS_SecOC_00033] [SWS_SecOC_00034] [SWS_SecOC_00035] [SWS_SecOC_00036] [SWS_SecOC_00037] [SWS_SecOC_00040] [SWS_SecOC_00042] [SWS_SecOC_00046] [SWS_SecOC_00057] [SWS_SecOC_00058] [SWS_SecOC_00106] [SWS_SecOC_00146] [SWS_SecOC_00157] [SWS_SecOC_00161] [SWS_SecOC_00219] [SWS_SecOC_00230] [SWS_SecOC_00231] [SWS_SecOC_00243] [SWS_SecOC_00261] [SWS_SecOC_00262] [SWS_SecOC_91003] [SWS_SecOC_91004]
[SRS_SecOC_00007]	Verification retry by the receiver	[SWS_SecOC_00047] [SWS_SecOC_00094] [SWS_SecOC_00234] [SWS_SecOC_00235] [SWS_SecOC_00236] [SWS_SecOC_00237] [SWS_SecOC_00238] [SWS_SecOC_00239] [SWS_SecOC_00240] [SWS_SecOC_00241] [SWS_SecOC_00242] [SWS_SecOC_00243]
[SRS_SecOC_00010]	Communication security is available for all communication paradigms of AUTOSAR	[SWS_SecOC_00060] [SWS_SecOC_00061] [SWS_SecOC_00062] [SWS_SecOC_00063] [SWS_SecOC_00064] [SWS_SecOC_00065] [SWS_SecOC_00066] [SWS_SecOC_00067] [SWS_SecOC_00068] [SWS_SecOC_00069] [SWS_SecOC_00070] [SWS_SecOC_00071] [SWS_SecOC_00072] [SWS_SecOC_00073] [SWS_SecOC_00074] [SWS_SecOC_00075] [SWS_SecOC_00078] [SWS_SecOC_00079] [SWS_SecOC_00080] [SWS_SecOC_00081] [SWS_SecOC_00082] [SWS_SecOC_00083] [SWS_SecOC_00084] [SWS_SecOC_00085] [SWS_SecOC_00086] [SWS_SecOC_00088] [SWS_SecOC_00150] [SWS_SecOC_00253] [SWS_SecOC_00254] [SWS_SecOC_00257] [SWS_SecOC_00258] [SWS_SecOC_00259] [SWS_SecOC_00267] [SWS_SecOC_00268]

▽

△

Requirement	Description	Satisfied by
[SRS_SecOC_00012]	Support of Automotive BUS Systems	[SWS_SecOC_00060] [SWS_SecOC_00061] [SWS_SecOC_00062] [SWS_SecOC_00063] [SWS_SecOC_00064] [SWS_SecOC_00065] [SWS_SecOC_00066] [SWS_SecOC_00067] [SWS_SecOC_00068] [SWS_SecOC_00069] [SWS_SecOC_00070] [SWS_SecOC_00071] [SWS_SecOC_00072] [SWS_SecOC_00073] [SWS_SecOC_00074] [SWS_SecOC_00075] [SWS_SecOC_00078] [SWS_SecOC_00079] [SWS_SecOC_00080] [SWS_SecOC_00081] [SWS_SecOC_00082] [SWS_SecOC_00083] [SWS_SecOC_00084] [SWS_SecOC_00085] [SWS_SecOC_00086] [SWS_SecOC_00088] [SWS_SecOC_00113] [SWS_SecOC_00124] [SWS_SecOC_00125] [SWS_SecOC_00126] [SWS_SecOC_00127] [SWS_SecOC_00128] [SWS_SecOC_00129] [SWS_SecOC_00130] [SWS_SecOC_00150] [SWS_SecOC_00152] [SWS_SecOC_00181] [SWS_SecOC_00253] [SWS_SecOC_00254] [SWS_SecOC_00257] [SWS_SecOC_00258] [SWS_SecOC_00259] [SWS_SecOC_00267] [SWS_SecOC_00268] [SWS_SecOC_00270] [SWS_SecOC_91009] [SWS_SecOC_91010]
[SRS_SecOC_00013]	Support for end-to-end and point-to-point protection	[SWS_SecOC_00060] [SWS_SecOC_00061] [SWS_SecOC_00062] [SWS_SecOC_00063] [SWS_SecOC_00064] [SWS_SecOC_00065] [SWS_SecOC_00066] [SWS_SecOC_00067] [SWS_SecOC_00068] [SWS_SecOC_00069] [SWS_SecOC_00070] [SWS_SecOC_00071] [SWS_SecOC_00072] [SWS_SecOC_00073] [SWS_SecOC_00074] [SWS_SecOC_00075] [SWS_SecOC_00078] [SWS_SecOC_00079] [SWS_SecOC_00080] [SWS_SecOC_00081] [SWS_SecOC_00082] [SWS_SecOC_00083] [SWS_SecOC_00084] [SWS_SecOC_00085] [SWS_SecOC_00086] [SWS_SecOC_00088] [SWS_SecOC_00150] [SWS_SecOC_00253] [SWS_SecOC_00254] [SWS_SecOC_00257] [SWS_SecOC_00258] [SWS_SecOC_00259] [SWS_SecOC_00268]
[SRS_SecOC_00017]	PDU security information override	[SWS_SecOC_00119] [SWS_SecOC_00122] [SWS_SecOC_00142] [SWS_SecOC_00991]
[SRS_SecOC_00020]	Security operational information persistency	[SWS_SecOC_00161]
[SRS_SecOC_00021]	Transmitted PDU authentication failure handling	[SWS_SecOC_00002] [SWS_SecOC_00076] [SWS_SecOC_00087] [SWS_SecOC_00151] [SWS_SecOC_00214] [SWS_SecOC_00215] [SWS_SecOC_00216] [SWS_SecOC_00217] [SWS_SecOC_00225] [SWS_SecOC_00226] [SWS_SecOC_00227] [SWS_SecOC_00228] [SWS_SecOC_00229] [SWS_SecOC_91002] [SWS_SecOC_91012] [SWS_SecOC_91013]

▽

△

Requirement	Description	Satisfied by
[SRS_SecOC_00022]	Received PDU verification failure handling	[SWS_SecOC_00047] [SWS_SecOC_00048] [SWS_SecOC_00050] [SWS_SecOC_00087] [SWS_SecOC_00121] [SWS_SecOC_00141] [SWS_SecOC_00148] [SWS_SecOC_00149] [SWS_SecOC_00160] [SWS_SecOC_00214] [SWS_SecOC_00215] [SWS_SecOC_00216] [SWS_SecOC_00236] [SWS_SecOC_00237] [SWS_SecOC_00238] [SWS_SecOC_00239] [SWS_SecOC_00240] [SWS_SecOC_00241] [SWS_SecOC_00248] [SWS_SecOC_00256] [SWS_SecOC_00271] [SWS_SecOC_00272] [SWS_SecOC_91002] [SWS_SecOC_91012]
[SRS_SecOC_00025]	Authentication and verification processing time	[SWS_SecOC_00173] [SWS_SecOC_00174] [SWS_SecOC_00175] [SWS_SecOC_00178] [SWS_SecOC_00179] [SWS_SecOC_00180]
[SRS_SecOC_00026]	Capability to transmit data and authentication information separately	[SWS_SecOC_00201] [SWS_SecOC_00202] [SWS_SecOC_00203] [SWS_SecOC_00204] [SWS_SecOC_00205] [SWS_SecOC_00206] [SWS_SecOC_00207] [SWS_SecOC_00208] [SWS_SecOC_00277]
[SRS_SecOC_00028]	Properly match up data and authentication information when verifying	[SWS_SecOC_00203] [SWS_SecOC_00209] [SWS_SecOC_00210] [SWS_SecOC_00211]
[SRS_SecOC_00029]	Flexible freshness construction	[SWS_SecOC_00219] [SWS_SecOC_00220] [SWS_SecOC_00221] [SWS_SecOC_00222] [SWS_SecOC_00223] [SWS_SecOC_00224] [SWS_SecOC_00225] [SWS_SecOC_00226] [SWS_SecOC_00227] [SWS_SecOC_00228] [SWS_SecOC_00229] [SWS_SecOC_00230] [SWS_SecOC_00231] [SWS_SecOC_00232] [SWS_SecOC_00233] [SWS_SecOC_00234] [SWS_SecOC_00235] [SWS_SecOC_00236] [SWS_SecOC_00237] [SWS_SecOC_00238] [SWS_SecOC_00239] [SWS_SecOC_00240] [SWS_SecOC_00241] [SWS_SecOC_00242] [SWS_SecOC_00243] [SWS_SecOC_00244] [SWS_SecOC_00245] [SWS_SecOC_00246] [SWS_SecOC_00247] [SWS_SecOC_00248] [SWS_SecOC_00249] [SWS_SecOC_00250] [SWS_SecOC_00256] [SWS_SecOC_91014] [SWS_SecOC_91015] [SWS_SecOC_91016] [SWS_SecOC_91021] [SWS_SecOC_91022]
[SRS_SecOC_00032]	Interaction decoupling between upper and lower layer modules	[SWS_SecOC_00252] [SWS_SecOC_00255]
[SWS_BSW_00242]	Access to *Meta Data*	[SWS_SecOC_00212]

Table 6.1: Requirements Tracing

7 Functional specification

Authentication and integrity protection of sensitive data is necessary to protect correct and safe functionality of the vehicle systems - this ensures that received data comes from the right ECU and has the correct value.

The SecOC module aims for resource-efficient and practicable authentication mechanisms of sensitive data on the level of PDUs. The approach proposed in this specification generally supports the use of symmetric and asymmetric methods for authenticity and integrity protection. Both methods roughly aim at the same goal and show major similarities in the concept, but there are also some differences due to differing technical properties of the underlying primitives. In addition, the commonly used terms for Authenticator are different. In general, the term Message Authentication Code (MAC) is used for symmetric approaches while the term signature or digital signature refers to asymmetric approaches having different properties and constraints.

In order to ease presentation and improve legibility, the following approach is taken: The subsequent section describes the technical approach using symmetric mechanisms in some detail. Here also the common terms for symmetric primitives are used. The adaptations that need to be done in case of an asymmetric approach are separately given in section 7.1.4.

7.1 Specification of the security solution

The SecOC module as described in this document provides functionality necessary to verify the authenticity and freshness of PDU based communication between ECUs within the vehicle architecture. The approach requires both the sending ECU and the receiving ECU to implement a SecOC module. Both SecOC modules are integrated providing the upper and lower layer PduR APIs on the sender and receiver side. The SecOC modules on both sides generally interact with the PduR module.

To provide message freshness, the SecOC module on the sending and receiving side get freshness from an external Freshness Manager for each uniquely identifiable Secured I-PDU, i.e. for each secured communication link.

On the sender side, the SecOC module creates a Secured I-PDU by adding authentication information to the outgoing Authentic I-PDU. The authentication information comprises of an Authenticator (e.g. Message Authentication Code) and optionally a Freshness Value. Regardless if the Freshness Value is or is not included in the Secure I-PDU payload, the Freshness Value is considered during generation of the Authenticator. When using a Freshness Counter instead of a Timestamp, the Freshness Counter should be incremented by the Freshness Manager prior to providing the authentication information to the receiver side.

On the receiver side, the SecOC module checks the freshness and authenticity of the Authentic I-PDU by verifying the authentication information that has been appended by the sending side SecOC module. To verify the authenticity and freshness of an Authen-

tic I-PDU, the Secured I-PDU provided to the receiving side SecOC should be the same Secured I-PDU provided by the sending side SecOC and the receiving side SecOC should have knowledge of the Freshness Value used by the sending side SecOC during creation of the Authenticator.

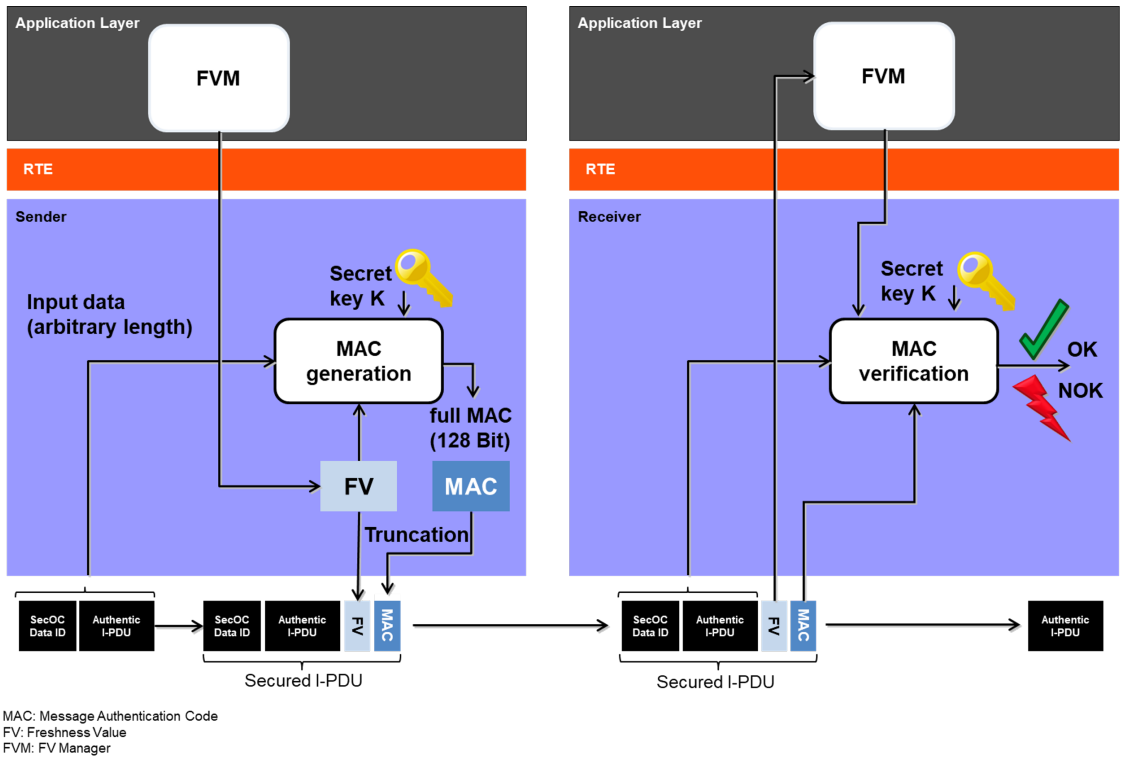

Figure 7.1: Simplified View of Message Authentication and Freshness Verification flow

The main purpose of the SecOC module is the realization of the security functionality described throughout this specification.

7.1.1 Basic entities of the security solution

7.1.1.1 Authentic I-PDU and Secured I-PDU

The term Authentic I-PDU refers to an AUTOSAR I-PDU that requires protection against unauthorized manipulation and replay attacks.

The payload of a Secured I-PDU consists of the Authentic I-PDU and an Authenticator (e.g. Message Authentication Code). The payload of a Secured I-PDU may optionally include the Freshness Value used to create the Authenticator (e.g. MAC). The order in which the contents are structured in the Secured I-PDU is compliant with 7.2.

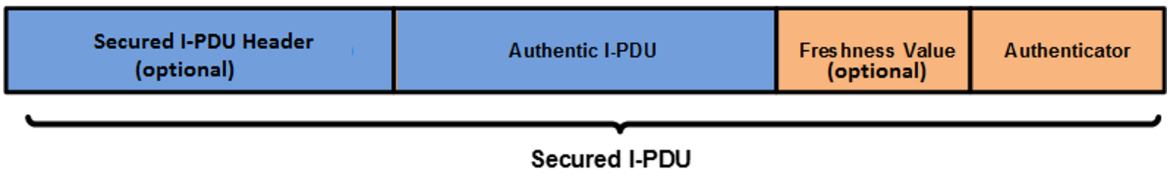

Figure 7.2: Secured I-PDU contents

The length of the Authentic I-PDU, the Freshness Value and the Authenticator within a Secured I-PDU may vary from one uniquely indefinable Secured I-PDU to another.

The Authenticator (e.g. MAC) refers to a unique authentication data string generated using a Key, Data Identifier of the Secured I-PDU, Authentic Payload, and Freshness Value. The Authenticator provides a high level of confidence that the data in an Authentic I-PDU is generated by a legitimate source and is provided to the receiving ECU at the time in which it is intended for.

Depending on the authentication algorithm(parameter `SecOCTxAuthServiceConfigRef` or `SecOCRxAuthServiceConfigRef`) used to generate the Authenticator, it may be possible to truncate the resulting Authenticator (e.g. in case of a MAC) generated by the authentication algorithm. Truncation may be desired when the message payload is limited in length and does not have sufficient space to include the full Authenticator.

The Authenticator length contained in a Secured I-PDU (parameter `SecOCAuthInfoTruncLength`) is specific to a uniquely identifiable Secured I-PDU. This allows provision of flexibility across the system (i.e. two independent unique Secured I-PDUs may have different Authenticator lengths included in the payload of the Secure I-PDU) by providing fine grain configuration of the MAC truncation length for each Secured I-PDU.

If truncation is possible, the Authenticator should only be truncated down to the most significant bits of the resulting Authenticator generated by the authentication algorithm. Figure 7.3 shows an example of the truncation of the Authenticator and the Freshness Values respecting the parameter `SecOCFreshnessValueTruncLength` and `SecOCAuthInfoTruncLength`.

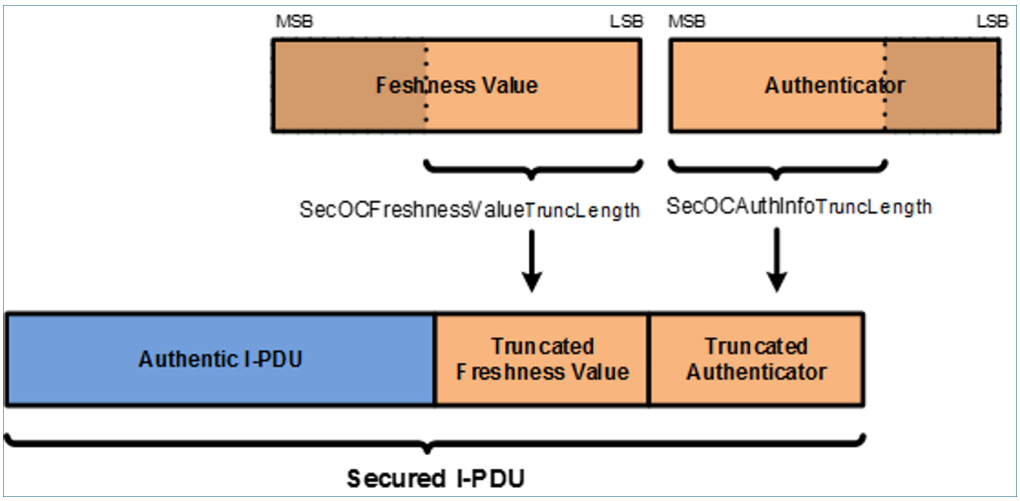

Figure 7.3: An example of Secured I-PDU contents with truncated Freshness Counter and truncated Authenticator (without Secured I-PDU Header)

Note: For the resource constraint embedded use case with static participants, we propose using Message Authentication Codes (MACs) as a basis for authentication (e.g. a CMAC [6] based on AES [7] with an adequate key length).

Note: In case a MAC is used, it is possible to transmit and compare only parts of the MAC. This is known as MAC truncation. However, this results in a lower security level at least for forgery of single MACs. While we propose to always use a key length of at least 128 bit, a MAC truncation can be beneficial. Of course, the actual length of the MAC for each use case has to be chosen carefully. For some guidance, we refer to appendix A of [6]. In general, MAC sizes of 64 bit and above are considered to provide sufficient protection against guessing attacks by NIST. Depending on the use case, different MAC sizes can be appropriate, but this requires careful judgment by a security expert.

[SWS_SecOC_00011]

Upstream requirements: SRS_SecOC_00006

⌈All SecOC data (e.g. Freshness Value, Authenticator, Data Identifier, SecOC message link data,...) that is directly or indirectly transmitted to the other side of a communication link shall be encoded in Big Endian byte order so that each SecOC module interprets the data in the same way.⌋

[SWS_SecOC_00261]

Upstream requirements: SRS_SecOC_00006

⌈The Secured I-PDU Header shall indicate the length of the Authentic I-PDU in bytes. The length of the Header shall be configurable by the parameter `SecOCAuthPduHeaderLength`.

Note: the SecOC supports combined usage of authentication data in a separate message (secured PDU collection) and Secured I-PDU Header. Also the SecOC covers dynamic length Authentic I-PDU.⌋

7.1.1.2 Data covered by Authenticator

The data, on which the Authenticator is calculated, consists of the Data Identifier of the Secured I-PDU (parameter SecOCDataId), Authentic I-PDU data, and the Complete Freshness Value. These are concatenated together respectively to make up the bit array that is passed into the authentication algorithm for Authenticator generation/verification.

DataToAuthenticator = Data Identifier | secured part of the Authentic I-PDU | Complete Freshness Value

Note: "|" denotes concatenation

7.1.1.3 Freshness Values

Each Secured I-PDU is configured with at least one Freshness Value. The Freshness Value refers to a monotonic counter that is used to ensure freshness of the Secured I-PDU. Such a monotonic counter could be realized by means of individual message counters, called Freshness Counter, or by a time stamp value called Freshness Timestamp. Freshness Values are to be derived from a Freshness Manager.

[SWS_SecOC_00094]

Upstream requirements: SRS_SecOC_00002, SRS_SecOC_00007

⌈If the parameter SecOCFreshnessValueTruncLength is configured to a smaller length than the actual freshness value, SecOC shall include only the least significant bits of the freshness value up to SecOCFreshnessValueTruncLength within the secured I-PDU.

If the parameter SecOCFreshnessValueTruncLength is configured to 0, the freshness value shall not be included in the secured I-PDU.⌋

Note: The larger number of bits of the complete Freshness Value included in the authenticated message payload results in a larger window where the receiver remains synchronized with the transmitters Freshness Value without executing a synchronization strategy.

Note: When including part of the Freshness Value in the authenticated message payload, the Freshness Value is referred to as two parts, the most significant bits and the least significant bits. The part of the counter included in the Secured I-PDU payload is

referred to as the least significant bits of the Freshness Value and the remaining part of the counter is referred to as the most significant bits of the Freshness Value.

[SWS_SecOC_00219]

Upstream requirements: SRS_SecOC_00006, SRS_SecOC_00029

⌈If SecOCUseAuthDataFreshness is set to TRUE, SecOC shall use a part of the Authentic I-PDU as freshness. In this case, SecOCAuthDataFreshnessStartPosition determines the start position in bits of the freshness inside the Authentic I-PDU and SecOCAuthDataFreshnessLen determines its length in bits.⌋

Note: This allows reusing existing freshness values from the payload which are guaranteed to be unique within the validity period of a Freshness Timestamp, e.g. a 4 bit E2E counter. In this case SecOC does not need to generate any additional counter values.

Example:

If SecOCUseAuthDataFreshness is set to TRUE, SecOCAuthDataFreshnessStartPosition is set to '11' and SecOCAuthDataFreshnessLen is set to '4', the following part of the PDU would be extracted:

Byte index of the PDU	0	1	...
Start bit numbering scheme	7 6 5 4 3 2 1 0	15 14 13 12 **11** **10 9 8**	...

For a PDU "AB CD" (hex), the authentic data freshness would be "1101" (bin).

[SWS_SecOC_00220]

Upstream requirements: SRS_SecOC_00029

⌈The Freshness Manager provides or receives freshness information in interface functions as byte arrays. The freshness is always aligned to the MSB of the first byte in the array. The 15th bit of the freshness is the MSB of the 2nd byte and so on. Unused bits of the freshness array must be set to 0. The associated length information must be given in bits.⌋

Example:

The 10-bit freshness "001101011" (bin) can be located in a 2 byte array and corresponds to the value: "35 80" (hex). The length value is 10.

[SWS_SecOC_00221]

Upstream requirements: SRS_SecOC_00029

⌈If SecOCQueryFreshnessValue = CFUNC AND SecOCProvideTxTruncatedFreshnessValue= TRUE for a PDU configuration, the SecOC calls the interface func-

tion SecOC_GetTxFreshnessTruncData whenever the DataToAuthenticator is constructed for the respective PDU.⌋

[SWS_SecOC_00222]

Upstream requirements: SRS_SecOC_00029

⌈If SecOCQueryFreshnessValue = CFUNC AND SecOCProvideTxTruncatedFreshnessValue= FALSE for a PDU configuration, the SecOC calls the interface function SecOC_GetTxFreshness whenever the DataToAuthenticator is constructed for the respective PDU.⌋

[SWS_SecOC_00223]

Upstream requirements: SRS_SecOC_00029

⌈If SecOCQueryFreshnessValue = RTE AND SecOCProvideTxTruncatedFreshnessValue= TRUE for a PDU configuration, the SecOC calls the service operation FreshnessManagement.GetTxFreshnessTruncData whenever the DataToAuthenticator is constructed for the respective PDU.⌋

[SWS_SecOC_00224]

Upstream requirements: SRS_SecOC_00029

⌈If SecOCQueryFreshnessValue = RTE AND SecOCProvideTxTruncatedFreshnessValue= FALSE for a PDU configuration, the SecOC calls the service operation FreshnessManagement.GetTxFreshness whenever the DataToAuthenticator is constructed for the respective PDU.⌋

[SWS_SecOC_00225]

Upstream requirements: SRS_SecOC_00021, SRS_SecOC_00029

⌈For every transmission request that is queued to SecOC an authentication build counter shall be maintained.⌋

[SWS_SecOC_00226]

Upstream requirements: SRS_SecOC_00005, SRS_SecOC_00021, SRS_SecOC_00029

⌈Upon the initial processing of a transmission request of a secured I-PDU SecOC shall set the authentication build counter to 0.⌋

[SWS_SecOC_00227]

Upstream requirements: SRS_SecOC_00021, SRS_SecOC_00029

⌈If either the query of the freshness function (e.g. SecOC_GetTxFreshness()) returns E_BUSY or the calculation of the authenticator (e.g. Csm_MacGenerate()) returns E_BUSY, QUEUE_FULL or any other recoverable error, the authentication build counter shall be incremented.⌋

Note: The return value E_NOT_OK is not considered as a recoverable error.

[SWS_SecOC_00228]

Upstream requirements: SRS_SecOC_00021, SRS_SecOC_00029

⌈If building the authentication has failed and the authentication build counter has not yet reached the configuration value SecOCAuthenticationBuildAttempts, the freshness attempt and authenticator calculation shall be retried in the next call to the Tx main function.⌋

[SWS_SecOC_00229]

Upstream requirements: SRS_SecOC_00021, SRS_SecOC_00029

⌈If the authentication build counter has reached the configuration value SecOCAuthenticationBuildAttempts, or the query of the freshness function returns E_NOT_OK or the calculation of the authenticator has returned a non-recoverable error such as returning E_NOT_OK or KEY_FAILURE, the SecOC module shall use RTE for all the bytes of Freshness Value and Authenticator to build the Authentication Information if sending RTE is enabled by service SecOC_SendDefaultAuthenticationInformation . If sending RTE is not enabled, the SecOc module shall remove the Authentic I-PDU from its internal buffer and cancel the transmission request.⌋

Note:

Example:

SecOCFreshnessValueTxLength = 4bits

SecOCAuthInfoTxLength = 20 bits

SecOCDefaultAuthenticatorValue = 0xA5

The resulting default Authentication Information within the secured PDU would be 0x05 (Truncated Freshness Value) | 0xA5 0xA5 0xA0 (Truncated Authenticator). "|" denotes concatenation.

[SWS_SecOC_00230]

Upstream requirements: SRS_SecOC_00003, SRS_SecOC_00006, SRS_SecOC_00029

⌈If SecOCQueryFreshnessValue = CFUNC AND SecOCProvideTxTruncatedFreshnessValue= TRUE for a PDU configuration, SecOC calls a function named SecOC_GetTxFreshnessTruncData, to get the current freshness for TX messages.⌋

[SWS_SecOC_00231]

Upstream requirements: SRS_SecOC_00003, SRS_SecOC_00006, SRS_SecOC_00029

⌈If SecOCQueryFreshnessValue = CFUNC AND SecOCProvideTxTruncated-FreshnessValue= FALSE for a PDU configuration, SecOC calls a function named SecOC_GetTxFreshness, to get the current freshness for TX messages.⌋

[SWS_SecOC_00232]

Upstream requirements: SRS_SecOC_00002, SRS_SecOC_00003, SRS_SecOC_00029

⌈If SecOCQueryFreshnessValue = CFUNC for a PDU configuration, SecOC calls a function with the signature described in [SWS_SecOC_91005] to indicate that the Secured I-PDU has been successfully initiated for transmission.⌋

Note: It is not intended, that this function is called after the message has appeared on the bus. It is considered to be more secure calling this function after the successful transmission request to the PduR.

[SWS_SecOC_00233]

Upstream requirements: SRS_SecOC_00002, SRS_SecOC_00029

⌈If SecOCQueryFreshnessValue = RTE for a PDU configuration, SecOC calls the service operation FreshnessManagement.SPduTxConfirmation to indicate that the Secured I-PDU has been successfully initiated for transmission.⌋

[SWS_SecOC_00234]

Upstream requirements: SRS_SecOC_00007, SRS_SecOC_00029

⌈For every processed secured I-PDU within SecOC an authentication build counter and an authentication verify attempt counter shall be maintained.⌋

[SWS_SecOC_00235]

Upstream requirements: SRS_SecOC_00005, SRS_SecOC_00007, SRS_SecOC_00029

⌈Upon the initial processing of a received secured I-PDU, the authentication build counter and the authentication verify attempt counter shall be set to 0.⌋

[SWS_SecOC_00236]

Upstream requirements: SRS_SecOC_00007, SRS_SecOC_00022, SRS_SecOC_00029

⌈If the query of the freshness function (e.g. SecOC_GetRxFreshness()) returns E_-BUSY the authentication build counter shall be incremented and no attempt for verification of authentication shall be executed.⌋

[SWS_SecOC_00237]

Upstream requirements: SRS_SecOC_00007, SRS_SecOC_00022, SRS_SecOC_00029

⌈If the verification of the authenticator (e.g. `Csm_MacVerify()`) returns `E_BUSY`, `QUEUE_FULL` or any other recoverable error, the authentication build counter shall be incremented.⌋

Note: The return value `E_NOT_OK` is not considered as a recoverable error.

[SWS_SecOC_00238]

Upstream requirements: SRS_SecOC_00007, SRS_SecOC_00022, SRS_SecOC_00029

⌈If the authentication build attempts have failed and the authentication build counter has not yet reached the configuration value `SecOCAuthenticationBuildAttempts`, the freshness attempt and the authenticator verification shall be retried in the next call to the Rx main function.⌋

[SWS_SecOC_00239]

Upstream requirements: SRS_SecOC_00007, SRS_SecOC_00022, SRS_SecOC_00029

⌈If the verification of the authenticator could be successfully executed but the verification failed (e.g. the MAC verification has failed or the key was invalid), the authentication verify attempt counter shall be incremented and the authentication build counter shall be set to 0.⌋

Note: Resetting the authentication build counter shall prevent to drop the authentication process too early even though authentication verify attempts are still possible.

[SWS_SecOC_00240]

Upstream requirements: SRS_SecOC_00007, SRS_SecOC_00022, SRS_SecOC_00029

⌈If the authentication build counter has reached the configuration value `SecOCAuthenticationBuildAttempts` the SecOC module shall remove the Authentic I-PDU from its internal buffer and shall drop the received message. The `SecOC_VerificationResultType` shall be set to `SECOC_AUTHENTICATIONBUILDFAILURE`.

if `SecOC_VerifyStatusOverride` is used, the verification result and I-PDU are handled according to overrideStatus value.⌋

[SWS_SecOC_00256]

Upstream requirements: SRS_BSW_00385, SRS_SecOC_00022, SRS_SecOC_00029

⌈If the query of the freshness function returns `E_NOT_OK` the SecOC module shall remove the Authentic I-PDU from its internal buffer and shall drop the received message. The `SecOC_VerificationResultType` shall be set to `SECOC_FRESHNESSFAILURE`.⌋

[SWS_SecOC_00241]

Upstream requirements: SRS_SecOC_00007, SRS_SecOC_00022, SRS_SecOC_00029

⌈If the authentication verify attempt counter has reached the configuration value `SecOCAuthenticationVerifyAttempts` or the verification of the authenticator has returned a non-recoverable error such as returning `E_NOT_OK` or `KEY_FAILURE`, the SecOC module shall remove the Authentic I-PDU from its internal buffer and shall drop the received message. The `SecOC_VerificationResultType` shall be set to `SECOC_VERIFICATIONFAILURE`.

If `SecOC_VerifyStatusOverride` is used, the verification result and I-PDU are handled according to overrideStatus value.⌋

Note: The sequence diagram in 9.4 illustrates this behavior.

[SWS_SecOC_00242]

Upstream requirements: SRS_SecOC_00007, SRS_SecOC_00029

⌈If the verification of the authenticator was successful, the `SecOC_VerificationResultType` shall be set to `SECOC_VERIFICATIONSUCCESS`.⌋

[SWS_SecOC_00243]

Upstream requirements: SRS_SecOC_00006, SRS_SecOC_00007, SRS_SecOC_00029

⌈The Freshness Management shall use the verification status callout function ([SWS_SecOC_00119]) to get the result of the verification of a secured I-PDU. This notification can be used as example to synchronize additional freshness attempts or can be used for counter increments.⌋

Note: SecOC allows to overwrite the status (see [SWS_SecOC_00142]). Therefore, care must be taken if the Freshness Management relies on the status callout while status overwrite function is also used. This can lead to conflicts in the Freshness Management and may lead to incorrect freshness values.

[SWS_SecOC_00244]

Upstream requirements: SRS_SecOC_00003, SRS_SecOC_00029

⌈If `SecOCQueryFreshnessValue` = `RTE` AND `SecOCUseAuthDataFreshness` = TRUE for a PDU configuration and the secured PDU is received completely, the SecOC calls the Rte service `FreshnessManagement.GetRxFreshnessAuthData` to query the current freshness. A part of the received PDU data are passed to this service operation as configured by the configuration `SecOCAuthDataFreshnessStartPosition` and `SecOCAuthDataFreshnessLen`.⌋

[SWS_SecOC_00245]

Upstream requirements: SRS_SecOC_00003, SRS_SecOC_00029

⌈If SecOCQueryFreshnessValue = RTE AND SecOCUseAuthDataFreshness = FALSE for a PDU configuration and the secured PDU is received completely, the SecOC calls the Rte service FreshnessManagement.GetRxFreshness to query the current freshness.⌋

[SWS_SecOC_00246]

Upstream requirements: SRS_SecOC_00003, SRS_SecOC_00029

⌈If SecOCQueryFreshnessValue = CFUNC AND SecOCUseAuthDataFreshness = TRUE for a PDU configuration and the secured PDU is received completely, the SecOC calls the interface function SecOC_GetRxFreshnessAuthData to query the current freshness. A part of the received PDU data are passed to this function as configured by the configuration SecOCAuthDataFreshnessStartPosition and SecOCAuthDataFreshnessLen.⌋

[SWS_SecOC_00247]

Upstream requirements: SRS_SecOC_00003, SRS_SecOC_00029

⌈If SecOCQueryFreshnessValue = CFUNC AND SecOCUseAuthDataFreshness = FALSE for a PDU configuration and the secured PDU is received completely, the SecOC calls the interface function SecOC_GetRxFreshness to query the current freshness.⌋

[SWS_SecOC_00248]

Upstream requirements: SRS_SecOC_00022, SRS_SecOC_00029

⌈If the Rx freshness request function returns E_NOT_OK, the verification of an Authentic I-PDU is considered to be failed and the authentication retry counter for this PDU shall be incremented. If the number of authentication attempts has reached SecOCAuthenticationVerifyAttempts, the SecOC module shall remove the Authentic I-PDU from its internal buffer. The failure SECOC_E_FRESHNESS_FAILURE shall be reported to the DET module.⌋

[SWS_SecOC_00249]

Upstream requirements: SRS_SecOC_00003, SRS_SecOC_00029

⌈If SecOCQueryFreshnessValue = CFUNC AND SecOCUseAuthDataFreshness = TRUE for a PDU configuration, SecOC queries a function named SecOC_GetRxFreshnessAuthData, to get the current freshness for RX messages.⌋

[SWS_SecOC_00250]

Upstream requirements: SRS_SecOC_00003, SRS_SecOC_00029

⌈If SecOCQueryFreshnessValue = CFUNC AND SecOCUseAuthDataFreshness = FALSE for a PDU configuration, SecOC queries a function named SecOC_GetRxFreshness, to get the current freshness for RX messages.⌋

7.1.2 Authentication of I-PDUs

[SWS_SecOC_00031]

Upstream requirements: SRS_SecOC_00006

⌈The creation of a Secured I-PDU and thus the authentication of an Authentic I-PDUconsists of the following six steps:

1. Prepare Secured I-PDU
2. Construct Data for Authenticator
3. Generate Authenticator
4. Construct Secured I-PDU
5. Increment Freshness Counter
6. Broadcast Secured I-PDU

⌋

[SWS_SecOC_00033]

Upstream requirements: SRS_SecOC_00006

⌈The SecOC module shall prepare the Secured I-PDU. During preparation,SecOC shall allocate the necessary buffers to hold the intermediate and final results of the authentication process.⌋

[SWS_SecOC_00034]

Upstream requirements: SRS_SecOC_00006

⌈The SecOC module shall construct the DataToAuthenticator, i.e. the data that is used to calculate the Authenticator. DataToAuthenticator is formed by concatenating the full 16 bit representation of the Data Id (parameter SecOCDataId), the secured part of the Authentic I-PDU and the complete Freshness Value corresponding to SecOCFreshnessValueId in the given order. The Data Id and the Freshness Value shall be encoded in Big Endian byte order for that purpose.⌋

[SWS_SecOC_00035]

Upstream requirements: SRS_SecOC_00006

⌈The SecOC module shall generate the Authenticator by passing `DataToAuthenticator`, length of `DataToAuthenticator` into the Authentication Algorithm corresponding to `SecOCTxAuthServiceConfigRef`.⌋

[SWS_SecOC_00036]

Upstream requirements: SRS_SecOC_00006

⌈The SecOC module shall truncate the resulting Authenticator down to the number of bits specified by `SecOCAuthInfoTruncLength`.⌋

[SWS_SecOC_00037]

Upstream requirements: SRS_SecOC_00006

⌈The SecOC module shall construct the Secured I-PDU by adding the Secured I-PDU Header (optional), the Freshness Value (optional) and the Authenticator to the Authentic I-PDU.

The scheme for the Secured I-PDU (includes the order in which the contents are structured in the Secured I-PDU) shall be compliant with below:

SecuredPDU = SecuredIPDUHeader (optional) | AuthenticIPDU | FreshnessValue [`SecOCFreshnessValueTruncLength`] (optional) | Authenticator [`SecOCAuthInfoTruncLength`]⌋

Note: The Freshness Counter and the Authenticator included as part of the Secured I-PDU may be truncated per configuration specific to the identifier of the Secured I-PDU. Also, Freshness Value may be a part of Authentic I-PDU (see [[SWS_SecOC_00219]]).

7.1.3 Verification of I-PDUs

[SWS_SecOC_00040]

Upstream requirements: SRS_SecOC_00006

⌈The verification of a Secured I-PDU consists of the following six steps:

- Parse Authentic I-PDU, Freshness Value and Authenticator
- Get Freshness Value from Freshness Manager
- Construct Data to Authentication
- Verify Authentication Information
- Send Confirmation to Freshness Manager

- Pass Authentic I-PDU to upper layer

⌋

[SWS_SecOC_00203]

Upstream requirements: SRS_SecOC_00026, SRS_SecOC_00028

⌈If `SecOCRxSecuredPduCollection` is used then SecOC shall not perform any verification until it has received both the Authentic I-PDU and Cryptographic I-PDU which make up the Secured I-PDU. Only after both have been received SecOC shall attempt to verify the resulting Secure I-PDU. If `SecOC_VerifyStatusOverride` is used, the verification result and I-PDU are handled according to overrideStatus value.⌋

Note: This applies to all instances when a Secured I-PDU is received by SecOC from the PduR, which happens in parts as described above when `SecOCRxSecuredPduCollection` is used. There is no further distinction made throughout this document to avoid duplication and clutter.

[SWS_SecOC_00211]

Upstream requirements: SRS_SecOC_00028

⌈If `SecOCRxSecuredPduCollection` is used then SecOC shall not attempt to verify the Secured I-PDU until it has received and buffered an Authentic I-PDU and Cryptographic I-PDU with matching Message Linker values. If `SecOC_VerifyStatusOverride` is used, the verification result and I-PDU are handled according to overrideStatus value.⌋

Note: If `SecOCUseMessageLink` has 0 multiplicity, it means `SecOCMessageLinkLen` is 0 and that Message Linker Values are always matching.

[SWS_SecOC_00042]

Upstream requirements: SRS_SecOC_00006

⌈Upon reception of a secured I-PDU, SecOC shall parse the Authentic I-PDU, the Freshness Value and the Authenticator from it.⌋

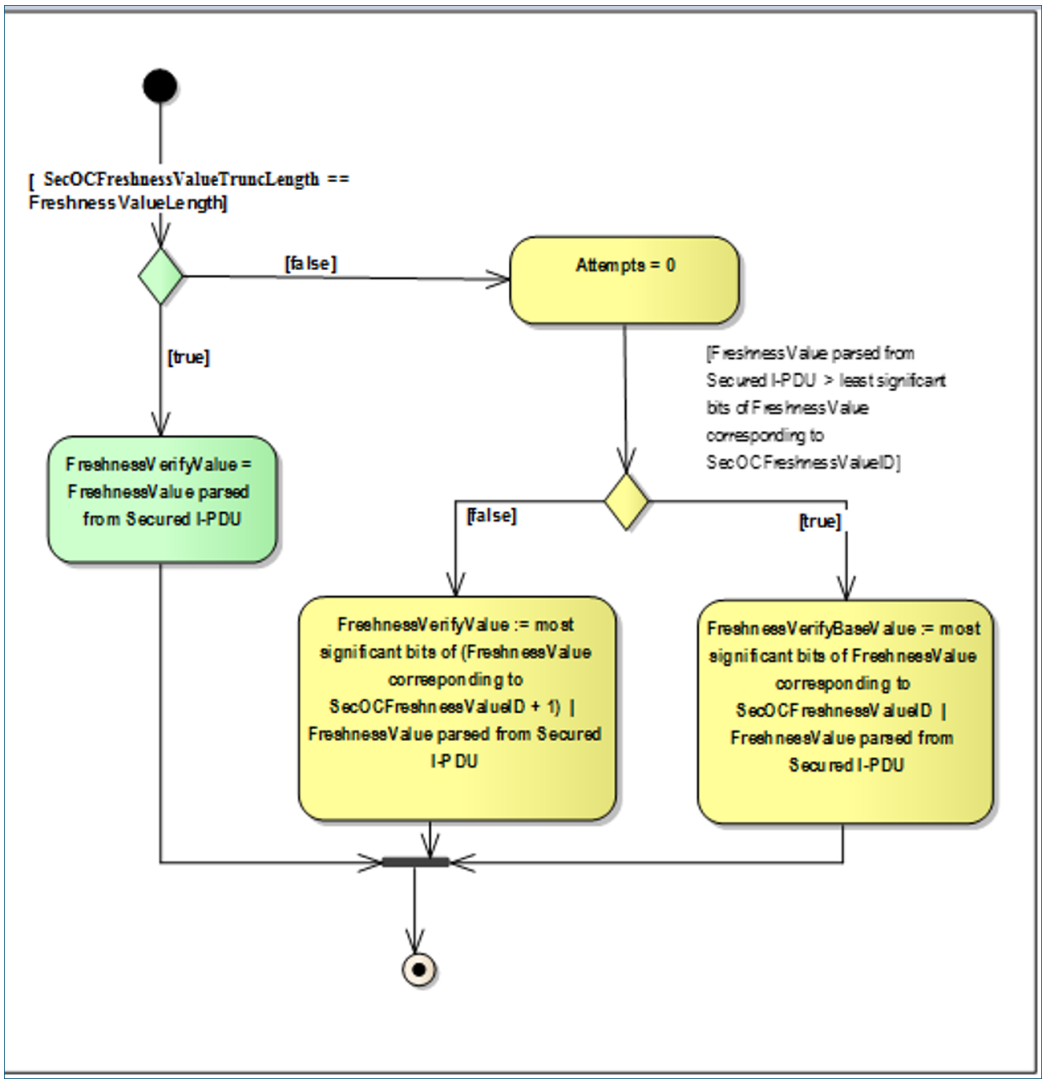

Figure 7.4: Construction of Freshness Value

[SWS_SecOC_00046]

Upstream requirements: SRS_SecOC_00006

⌈The SecOC module shall construct the data that is used to calculate the Authenticator (`DataToAuthenticator`) on the receiver side. This data is comprised of `SecOCDataId`| AuthenticIPDU | FreshnessVerifyValue⌋

[SWS_SecOC_00047]

Upstream requirements: SRS_SecOC_00002, SRS_SecOC_00007, SRS_SecOC_00022

⌈The SecOC module shall verify the Authenticator by passing `DataToAuthenticator`, length of `DataToAuthenticator`, the Authenticator parsed from Secured I-PDU, and `SecOCAuthInfoTruncLength` into the authentication algorithm corresponding to `SecOCRxAuthServiceConfigRef`.

The verification process is repeated as outlined in the sequence diagrams of this document.

If SecOC_VerifyStatusOverride is used, the verification result and I-PDU are handled according to overrideStatus value.⌋

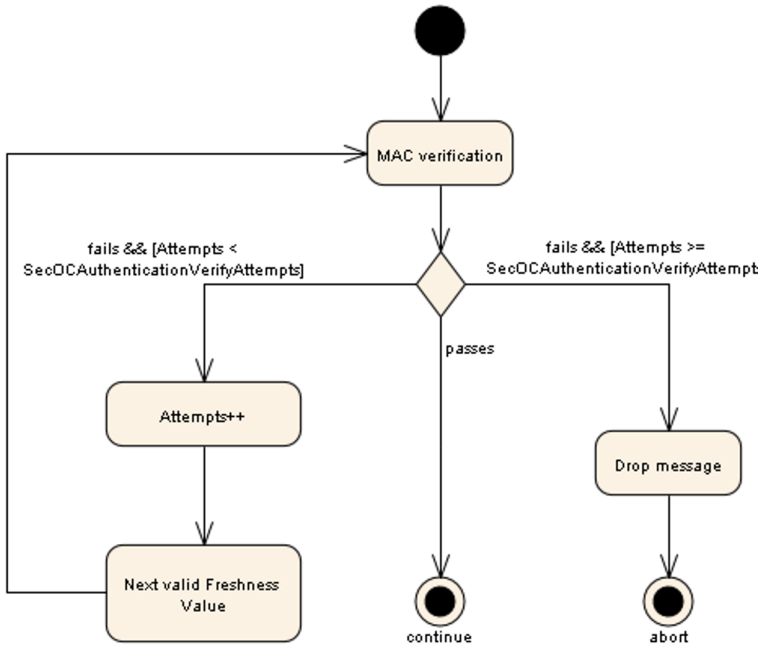

Figure 7.5: Verification of MAC

[SWS_SecOC_00048]

Upstream requirements: SRS_SecOC_00022

⌈The SecOC module shall report the verification status of the corresponding secured Rx-PDU as follows:

If SecOCRxPduProcessing/SecOCVerificationStatusPropagationMode is set to BOTH or FAILURE_ONLY, the verification status shall be served through the call out function SecOC_VerificationStatusCallout and the VerificationStatus interface according to its current configuration. No report will be provided if the configuration is set to NONE.⌋

Note: If the Freshness Manager requires the status of a secured PDU if it was verified successfully or not, e.g. to synchronize time or counter, then this status shall be taken from the VerificationStatus service provided by SecOC.

[SWS_SecOC_00271]

Upstream requirements: SRS_SecOC_00022

⌈The SecOC module shall report the verification status of the corresponding secured Rx-PDU as follows:

If `SecOCRxPduProcessing`/`SecOCClientServerVerificationStatusPropagationMode` is set to `BOTH` or `FAILURE_ONLY`, the verification status shall be served through the service interface `VerificationStatusIndication` according to its current configuration. No report will be provided if the configuration is set to `NONE`.⌋

[SWS_SecOC_00272]

Upstream requirements: SRS_SecOC_00022

⌈If the configuration item `SecOCGeneral`/`SecOCPropagateOnlyFinalVerificationStatus` is set to TRUE, then only the final status shall be reported. If this item is set to FALSE, then each individual verification status (the final one as well as all previous failed ones) shall be reported according to [SWS_SecOC_00048] and [SWS_SecOC_00271].⌋

7.1.3.1 Successful verification of I-PDUs

[SWS_SecOC_00050]

Upstream requirements: SRS_SecOC_00022

⌈If the verification of a Secured I-PDU was successful or the status override was set accordingly, the SecOC module shall pass the Authentic I-PDU to the upper layer communication modules using the lower layer interfaces of the PduR.⌋

7.1.4 Adaptation in case of asymmetric approach

Although this document consequently uses the terms and concepts from symmetric cryptography, the SecOC module can be configured to use both,symmetric as well as asymmetric cryptographic algorithms.In case of an asymmetric approach using digital signatures instead of the MAC-approach described throughout the whole document, some adaptations have to be made:

1. Instead of a shared secret between sender and (all) receivers, a key pair consisting of public key and secret key is used. The secret (or private) key is used by the sender to generate the signature, the corresponding public keys is used by (all) receiver(s) to verify the signature. The private key must not be feasibly computable from the public key and it shall not be assessable by the receivers.

2. In order to verify a message, the receiver needs access to the complete signature /output of the signature generation algorithm. Therefore, a truncation of the signature as proposed in the MAC case is NOT possible. The parameter `SecOCAuthInfoTruncLength` has to be set to the complete length of the signature.
3. The signature verification uses a different algorithm then the signature generation. So instead of "rebuilding" the MAC on receiver side and comparing it with the received (truncated) MAC as given above, the receiver / verifier performs the verification algorithm using the `DataToAuthenticator` (including full counter) and the signature as inputs and getting a Boolean value as output, determining whether the verification passed or failed.

7.2 Relationship to PduR

The SecOC module is arranged next to the PDU-Router in the layered architecture of AUTOSAR; see Figure 7.6.

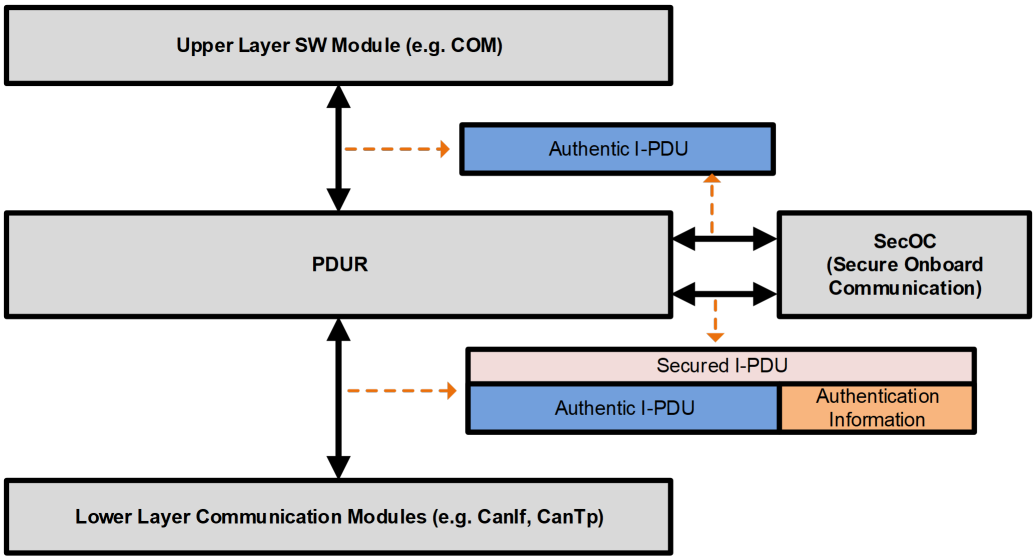

Figure 7.6: Transformation of an Authentic I-PDU in a Secured I-PDU by SecOC

[SWS_SecOC_00153]

Upstream requirements: SRS_BSW_00171

⌈The SecOC module shall be implemented so that no other modules depend on it and that it is possible to build a system without the SecOC module if it is not needed.⌋

[SWS_SecOC_00212]

Upstream requirements: SWS_BSW_00242

⌈SecOC shall ensure that MetaData received in an authentic PDU will be present unchanged in the corresponding secured PDU, and vice versa.⌋

7.3 Initialization

The SecOC module provides an initialization function (SecOC_Init) as defined in [SWS_SecOC_00106]. This function initializes all internal global variables and the buffers to store the SecOC I-PDUs and all intermediate results. The environment of the SecOC shall call SecOC_Init before calling any other function of the SecOC module except SecOC_GetVersionInfo. The implementer has to ensure that SECOC_E_UNINIT is returned in development mode in case an API function is called before the module is initialized.

For the I-PDU data transmission pathway through the SecOC module, a buffer is allocated inside the SecOC module. This buffer needs to be initialized because it might be transmitted before it has been fully populated with data by the upper layer of lower layer communication modules.

[SWS_SecOC_00054]

Upstream requirements: SRS_SecOC_00005

⌈Within SecOC_Init, the module shall initialize all internal global variables and the buffers of the SecOC I-PDUs.⌋

[SWS_SecOC_00269]

Upstream requirements: SRS_BSW_00101

⌈The AUTOSAR SecOC module shall fill not used areas of a transmitted Secured or a transmitted Cryptographic Pdu with a value determined by configuration parameter SecOCTxPduUnusedAreasDefault ([ECUC_SecOC_00101]) e.g. 0xFF.⌋

7.4 Authentication of outgoing PDUs

The term authentication describes the creation of a Secured I-PDU by adding Authentication Information to an Authentic I-PDU. This process is described in general terms in Section 7.1.2. This section refines the general description with respect to requirements arising from the integration with the PduR module considering different bus interfaces and transport protocols. In general, the interaction with the PduR module and the authentication of Authentic I-PDUs are organized according to the following scheme:

1. For each transmission request of an Authentic I-PDU, the upper layer communication module shall call the PduR module through PduR_<User:Up>Transmit.
2. The PduR routes this request to the SecOC module and calls SecOC_IfTransmit|SecOC_TpTransmit.
3. The SecOC module copies the Authentic I-PDU to its own memory and returns.

4. During the next scheduled call of its main function, the SecOC module creates the Secured I-PDU by calculating the Authentication Information and initiates the transmission of the Secured I-PDU by notifying the respective lower layer module via the PduR module.
5. Thereafter, the SecOC module takes the role of an upper layer communication module and thus serves all lower layer requests to provide information on or to copy data of the Secured I-PDU.
6. Finally, the confirmation of the successful or unsuccessful transmission of the Secured I-PDU are provided to the upper layer communication module as confirmation of the successful or unsuccessful transmission of the Authentic I-PDU

Note: For each Authentic I-PDU, the upper layer communication module shall be configured in such a way that it calls the PduR module as it normally does for a direct transmission request. In this case, the upper layer is decoupled from TriggerTransmit and TP behavior by means of the SecOC module.

To initiate the transmission of an Authentic I-PDU, the upper layer module always (and independent of the bus interface that is used for the concrete transmission) calls the PduR module through `PduR_<User:Up>Transmit`. The PduR routes this request to the SecOC module so that the SecOC module has immediate access to the Authentic I-PDU in the buffer of the upper layer communication module.

[SWS_SecOC_00252]

Upstream requirements: SRS_SecOC_00032

⌈The SecOC module shall copy the complete Authentic I-PDU to its internal memory before starting transmission of the corresponding Secured I-PDU.⌋

Note: This means there is no dependency between the IF/TP configuration of Up versus Lower PDU interfaces.

[SWS_SecOC_00201]

Upstream requirements: SRS_SecOC_00026

⌈If `SecOCTxSecuredPduCollection` is used, then SecOC shall transmit the Secured I-PDU as two messages: The original Authentic I-PDU and a separate Cryptographic I-PDU. The Cryptographic I-PDU shall contain all Authentication Information of the Secured I-PDU, so that the Authentic I-PDU and the Cryptographic I-PDU contain all information necessary to reconstruct the Secured I-PDU.⌋

Note: This applies to all instances when a Secured I-PDU is transmitted by SecOC to the PduR. There is no further distinction made throughout this document to avoid duplication and clutter.

[SWS_SecOC_00202]

Upstream requirements: SRS_SecOC_00026

⌈SecOC shall transmit an Authentic I-PDU and its corresponding Cryptographic I-PDU within the same main function cycle.⌋

[SWS_SecOC_00209]

Upstream requirements: SRS_SecOC_00028

⌈If `SecOCTxSecuredPduCollection` is used then SecOC shall repeat a part of the Authentic I-PDU inside the Cryptographic I-PDU as Message Linker and the Cryptographic I-PDU shall be constructed as

Cryptographic I-PDU =Authentication Data | Message Linker⌋

Note: "|" denotes concatenation.

[SWS_SecOC_00210]

Upstream requirements: SRS_SecOC_00028

⌈If `SecOCUseMessageLink` is used then SecOC shall use the value at bit position `SecOCMessageLinkPos` of length `SecOCMessageLinkLen` bits inside the Authentic I-PDU as the Message Linker.⌋

[SWS_SecOC_00270]

Upstream requirements: SRS_SecOC_00012

⌈If `SecOCTxSecuredPduCollection` is used, the SecOC shall forward the TxConfirmation to the upper layer if the `SecOC_TxConfirmation` was called for the Authentic I-PDU and the Cryptographic I-PDU. The reult parameter of the upper layer TxConfirmation call shall only be `E_OK` if the result parameters for both TxConfirmation calls were `E_OK`, Otherwise the result parameter shall be `E_NOT_OK`.⌋

[SWS_SecOC_00057]

Upstream requirements: SRS_SecOC_00006

⌈The SecOC module shall provide sufficient buffer capacities to store the incoming Authentic I-PDU, the outgoing Secured I-PDU and all intermediate data of the authentication process according to the process described in [SWS_SecOC_00031].⌋

[SWS_SecOC_00146]

Upstream requirements: SRS_SecOC_00006

⌈The SecOC module shall provide separate buffers for the Authentic I-PDU and the Secured I-PDU.⌋

[SWS_SecOC_00110]

Upstream requirements: SRS_BSW_00426

⌈Any transmission request from the upper layer communication module shall overwrite the buffer that contains the Authentic I-PDU without affecting the buffer of the respective Secured I-PDU.⌋

Thus, upper layer updates for Authentic I-PDUs could be processed without affecting ongoing transmission activities of Secured I-PDUs with the lower layer communication module.

[SWS_SecOC_00262]

Upstream requirements: SRS_SecOC_00006

⌈For a Tx Secured I-PDU with `SecOCAuthPduHeaderLength` > 0, the SecOC module shall add the Secured I-PDU Header to the Secured I-PDU with the length of the Authentic I-PDU within the Secured I-PDU, to handle dynamic Authentic I-PDU.

Note: Primary purpose of this Header is to indicate the position of Freshness Value and Authenticator in Secured I-PDUs with dynamic length Authentic I-PDU.Also some buses which cannot select arbitrary length of L-PDU (e.g. CAN FD and FlexRay) require this Header, because the position of Freshness Value and Authenticator is not always at the end of the Secured I-PDU, as lower layer modules (e.g. CanIf and FrIf) may add bus-specific padding bytes after processing at SecOC (then the L-PDU containing the Secured I-PDU with padding will be: Secured I-PDU = Secured I-PDU Header | Authentic I-PDU | Freshness Value | Authenticator | Bus-specific padding).⌋

7.4.1 Authentication during direct transmission

For transmission of an Authentic I-PDU using bus interfaces that allow ad-hoc transmission (e.g. CanIf), the PDU Router module triggers the transmit operation of the SecOC module for an Authentic I-PDU.In this case, the SecOC module prepares the creation of aSecured I-PDUon basis of the Authentic I-PDU by allocating internal buffer capacities and by copying the Authentic I-PDU to a local buffer location. Afterwards it returns from `SecOC_IfTransmit`|`SecOC_TpTransmit`.

[SWS_SecOC_00058]

Upstream requirements: SRS_SecOC_00006

⌈The SecOC module shall allocate internal buffer capacities to store the Authentic I-PDU and the Authentication Information in a consecutive memory location.⌋

The actual creation of the Secured I-PDU is processed during the next subsequent call of the scheduled main function. This includes calculating the Authentication Infor-

mation according to [SWS_SecOC_00031] and adding the Authentication Information (i.e. the Authenticator and the possibly truncated Freshness Value) consecutively to the buffer location directly behind the Authentic I-PDU. Thereafter, SecOC module triggers the transmission of the Secured I-PDU to the destination lower layer module by calling `PduR_SecOCTransmit` at the PduR.

[SWS_SecOC_00060]

Upstream requirements: SRS_SecOC_00010, SRS_SecOC_00012, SRS_SecOC_00013

⌈For transmission of Authentic I-PDUs using bus interfaces that allow ad-hoc transmission (e.g. CanIf), the SecOC module shall calculate the Authenticator in the scheduled main function according to the overall approach specified in [SWS_SecOC_00031].⌋

[SWS_SecOC_00061]

Upstream requirements: SRS_SecOC_00010, SRS_SecOC_00012, SRS_SecOC_00013

⌈For transmission of Authentic I-PDUs using bus interfaces that allow ad-hoc communication (e.g. CanIf), the SecOC module shall create the Secured I-PDU in the scheduled main function.⌋

[SWS_SecOC_00062]

Upstream requirements: SRS_SecOC_00010, SRS_SecOC_00012, SRS_SecOC_00013

⌈The SecOC module shall provide the complete Secured I-PDU for further transmission to the destination lower layer module by triggering `PduR_SecOCTransmit`.⌋

[SWS_SecOC_00063]

Upstream requirements: SRS_SecOC_00010, SRS_SecOC_00012, SRS_SecOC_00013

⌈If the PDU Router module notifies the SecOC module that the destination lower layer module has either confirmed the transmission of the Secured I-PDU or reported an error during transmission by calling `SecOC_TxConfirmation`|`SecOC_TpTxConfirmation`, the SecOC module shall pass the received result of the respective Authentic I-PDU to the upper layer module by calling `PduR_SecOCTxConfirmation`|`PduR_SecOCTpTxConfirmation`.⌋

[SWS_SecOC_00064]

Upstream requirements: SRS_SecOC_00010, SRS_SecOC_00012, SRS_SecOC_00013

⌈For transmission of Authentic I-PDUs using bus interfaces that allow ad-hoc communication (e.g. CanIf), the SecOC module shall free the buffer that contains the Secured I-PDU if `SecOC_TxConfirmation` is called for the Secured I-PDU.⌋

7.4.2 Authentication during triggered transmission

For transmission of an Authentic I-PDU using bus interfaces that allow triggered transmission (e.g. FrIf), the upper layer is configured in such a way that it calls the PduR module like it normally does for a direct transmission. Thus, the upper layer module immediately provides access to the Authentic I-PDU by providing the required buffer information through `PduR_<User:Up>Transmit`. The PduR forwards this transmission request to the SecOC module by calling `SecOC_IfTransmit`. Before the SecOC can provide data to the lower layer through the triggered transmission interface at least one previous call of `SecOC_IfTransmit` is required. If `SecOC_TriggerTransmit` is called and no data can be provided `E_NOT_OK` is returned.

Note: Authentication for triggered transmission is only supported, if the upper layer initiates the transmission by explicitly calling `PduR_<User:Up>Transmit` in before. Triggered transmission in mode AlwaysTransmit shall not be used.

In turn, the SecOC module allocates sufficient buffer capacities to store the Authentic I-PDU, the Secured I-PDU and all intermediate data of the authentication process. The SecOC module copies the Authentic I-PDU into its own buffer and returns (see [SWS_SecOC_00057], [SWS_SecOC_00058]).

The actual creation of the Secured I-PDU is processed during the subsequent call of the scheduled main function. This includes calculating the Authentication Information according to [SWS_SecOC_00031] and adding the Authentication Information (i.e. the Authenticator and the possibly truncated Freshness Value) consecutively to the buffer location directly behind the Authentic I-PDU. Thereafter, SecOC module triggers the transmission of the Secured I-PDU to the destination lower layer module by calling `PduR_SecOCTransmit` at the PduR.

[SWS_SecOC_00065]

Upstream requirements: SRS_SecOC_00010, SRS_SecOC_00012, SRS_SecOC_00013

⌈For transmission of Authentic I-PDUs using bus interfaces that allow triggered transmission (e.g. FrIf), the SecOC module shall calculate the Authenticator in the scheduled main function according to the overall approach specified in [SWS_SecOC_00031].⌋

[SWS_SecOC_00066]

Upstream requirements: SRS_SecOC_00010, SRS_SecOC_00012, SRS_SecOC_00013

⌈For transmission of Authentic I-PDUs using bus interfaces that allow triggered transmission (e.g. FrIf), the SecOC module shall create the Secured I-PDU in the scheduled main function.⌋

In the following, the SecOC module serves as a data provider for the subsequent transmission request from the lower layer module. Thus, the SecOC module holds the complete Secured I-PDU and acts as the upper layer module. The upper layer module does

not expect any further call back that request the copying of the Authentic I-PDU to the lower layer module.

[SWS_SecOC_00067]

Upstream requirements: SRS_SecOC_00010, SRS_SecOC_00012, SRS_SecOC_00013

⌈For transmission of Authentic I-PDUs using bus interfaces that allow triggered transmission (e.g. FrIf), the SecOC module shall indicate the transmission request for the complete Secured I-PDU by triggering `PduR_SecOCTransmit` at the PduR. The PduR is responsible to further process the request and to notify the respective lower layer module.⌋

The destination lower layer module calls `PduR_<User:Lo>TriggerTransmit` when it is ready to transmit the Secured I-PDU. PduR forwards this request to the SecOC module and the SecOC module copies the complete Secured I-PDU to the lower layer. Afterwards it returns.

Note: The SecOc module must not forward the trigger transmit call to the upper layer but takes itself the role of the upper layer and copies the complete Secured I-PDU to the lower layer.

[SWS_SecOC_00068]

Upstream requirements: SRS_SecOC_00010, SRS_SecOC_00012, SRS_SecOC_00013

⌈When `SecOC_TriggerTransmit` is called by the PduR module, the SecOC module shall copy the Secured I-PDU to the lower layer destination module.⌋

[SWS_SecOC_00150]

Upstream requirements: SRS_SecOC_00010, SRS_SecOC_00012, SRS_SecOC_00013

⌈When `SecOC_TriggerTransmit` is called by the PduR module and the SecOC module is not able to provide a Secured I-PDU to the lower layer (no Secured I-PDU available), the SecOC module shall return the call with `E_NOT_OK`.⌋

Finally, when the lower layer confirms the processing of the Secured I-PDU via `PduR_<User:Lo>TxConfirmation` (the result can be positive, if the PDU was successfully sent or negative if a transmission was not possible), the confirmation is forwarded to the SecOC module by calling `SecOC_TxConfirmation`. In turn, the SecOC module passes the result of the transmission process of the Authentic I-PDU at the PduR module so that the PduR module could forward the result via <Up>_TxConfirmation to the upper layer module which was the source of the original I-PDU (see [SWS_SecOC_00063]).

During triggered transmission, the update rates of the upper layer modules and the lower layer modules might be different. Thus, the lower layer module might request a new transmission of a Secured I-PDU while the upper layer has not updated the Authentic I-PDU. In this case, the SecOC module supports the repeated transmission

of the Authentic I-PDU by means of an updated Secure I-PDU. Thus, it has to preserve the Authentic I-PDU until the Secured I-PDU has been sent and its transmission has been confirmed by a means of SecOC_TxConfirmation. In this case, the SecOC module treats the existing Authentic I-PDU as new and re-authenticates it during the subsequent call to the SecOC_MainFunctionTx.

[SWS_SecOC_00069]

Upstream requirements: SRS_SecOC_00010, SRS_SecOC_00012, SRS_SecOC_00013

⌈For transmission of Authentic I-PDUs using bus interfaces that allow triggered transmission (e.g. FrIf) and the transmission of the Secured I-PDU was confirmed by SecOC_TxConfirmation (successfully sent), the SecOC module shall free the buffer that contain Authentication Information and preserve the buffer that contain the Authentic I-PDU. If the parameter SecOCReAuthenticateAfterTriggerTransmit is set to true, the Authentic I-PDU shall be treated as if it has been set by the upper layer and thus shall undergo a new authentication procedure with the subsequent call of the SecOC_MainFunctionTx. Otherwise no reauthentication of the Authentic I-PDU is required.⌋

7.4.3 Authentication during transport protocol transmission

For transmission of an Authentic I-PDU using transport protocol transmission (e.g. CanTP, FrTp), the PDU Router module triggers the transmit operation of the SecOC module for an Authentic I-PDU.In this case, the SecOC module prepares the creation of a Secured I-PDU on basis of the Authentic I-PDU by allocation internal buffer capacities and by copying the Authentic I-PDU to a local buffer location. Afterwards it returns from SecOC_IfTransmit|SecOC_TpTransmit.

The actual creation of the Secured I-PDU is processed during the next following call of the scheduled main function. This includes calculating the Authentication Information according to [SWS_SecOC_00031] and adding the Authentication Information(i.e. the Authenticator and the possibly truncated Freshness Value) consecutively to the buffer location directly behind the Authentic I-PDU.

[SWS_SecOC_00253]

Upstream requirements: SRS_SecOC_00010, SRS_SecOC_00012, SRS_SecOC_00013

⌈In case SecOCPduType is configured to SECOC_TPPDU, then function SecOC_TpTransmit shall trigger the transmit operation for an Authentic I-PDU.⌋

[SWS_SecOC_00254]

Upstream requirements: SRS_SecOC_00010, SRS_SecOC_00012, SRS_SecOC_00013

⌈After a transmit operation for `SecOCPduType` of `SECOC_TPPDU` was triggered, the SecOC shall instruct the upper layer to copy the next part of the I-PDU to a local SecOC buffer by calling `PduR_SecOCTpCopyTxData`.⌋

Note: The call to `PduR_SecOCTpCopyTxData` may happen in the context of `SecOC_TpTransmit`.

[SWS_SecOC_00070]

Upstream requirements: SRS_SecOC_00010, SRS_SecOC_00012, SRS_SecOC_00013

⌈For transmission of Authentic I-PDUs using transport protocol, the SecOC module shall calculate the Authenticator in the scheduled main function according to the overall approach specified in [SWS_SecOC_00031]. In case `SecOCPduType` is configured to `SECOC_TPPDU` the freshness value shall be retrieved as late as possible i.e. just in time when this part of the message will be transmitted next to the bus.⌋

Note: The late freshness value retrieval is necessary to have an up-to-date value for the case that the TP transmission took a while

[SWS_SecOC_00071]

Upstream requirements: SRS_SecOC_00010, SRS_SecOC_00012, SRS_SecOC_00013

⌈For transmission of Authentic I-PDUs using transport protocol, the SecOC module shall create the Secured I-PDU in the scheduled main function.⌋

Thereafter, SecOC module triggers the transmission of the Secured I-PDU to the destination lower layer module by calling `PduR_SecOCTpStartOfReception` at the PduR. Thus, it notifies the lower level module about its transmission request for the Secured I-PDU.

[SWS_SecOC_00072]

Upstream requirements: SRS_SecOC_00010, SRS_SecOC_00012, SRS_SecOC_00013

⌈For transmission of Authentic I-PDUs using transport protocol, the SecOC module shall indicate the transmission request for the complete Secured I-PDU by triggering `PduR_SecOCTransmit` at the PduR. The PduR is responsible to further process the request and to notify the respective lower layer module.⌋

In the following, the SecOC module serves as a data provider for the subsequent transmission request from the lower layer module. Thus, the SecOC module holds the complete Secured I-PDU and acts as the upper layer module. The upper layer module does

not expect any further call back that request the copying of the Authentic I-PDU to the lower layer module.

When the PduR iteratively polls the SecOC module by means of SecOC_CopyTxData to effectively transmit the Secured I-PDU to a lower layer module, the SecOC module copies the NPDUs for the Secured I-PDU to the lower layer transport protocol module.

[SWS_SecOC_00073]

Upstream requirements: SRS_SecOC_00010, SRS_SecOC_00012, SRS_SecOC_00013

⌈For transmission of Authentic I-PDUs using transport protocol, the SecOC module shall copy the NPDUs addressed by SecOC_CopyTxData into the buffer of the transport protocol module. After each copy process, it returns from SecOC_CopyTxData.⌋

Finally, when the lower layer confirms the processing of the Secured I-PDU via PduR_<User:Lo>TxConfirmation (the result can be positive, if the PDU was successfully sent or negative if a transmission was not possible), the result is forwarded to the SecOC module and the SecOC module in turn confirms the processing of the Authentic I-PDU, so that the PduR module could forward the result via <Up>_TxConfirmation to the upper layer.

[SWS_SecOC_00074]

Upstream requirements: SRS_SecOC_00010, SRS_SecOC_00012, SRS_SecOC_00013

⌈For transmission of Authentic I-PDUs using transport protocol and when the lower Layer either confirms the transmission of the Secured I-PDU or signals an error during transmission by calling SecOC_TpTxConfirmation, the SecOC module shall in turn pass the received result of the Authentic I-PDU either by PduR_SecOCIfTxConfirmation in case SecOCPduType is configured to SECOC_IFPDU or by PduR_SecOCTpTxConfirmation in case SecOCPduType is configured to SECOC_TPPDU.⌋

[SWS_SecOC_00075]

Upstream requirements: SRS_SecOC_00010, SRS_SecOC_00012, SRS_SecOC_00013

⌈For transmission of Authentic I-PDUs using transport protocol, the SecOC module shall free the buffer that contains the Secured I-PDU only, if SecOC_TpTxConfirmation is called for the Secured I-PDU.⌋

7.4.4 Error handling and cancelation of transmission

[SWS_SecOC_00076]

Upstream requirements: SRS_SecOC_00021

⌈If the upper layer module requests a cancelation of an ongoing transmission of the Authentic I-PDU by calling `SecOC_IfCancelTransmit`|`SecOC_TpCancelTransmit`, the SecOC module shall immediately inform the lower layer transport protocol module to cancel the ongoing transmission of the Secured I-PDU, stop all internal actions related to the Authentic I-PDU, and free all related buffers.⌋

[SWS_SecOC_00277]

Upstream requirements: SRS_SecOC_00026

⌈When `PduR_SecOCTransmit` returns anything but `E_OK`, the SecOC shall retry the failed transmission from the next main function up to `SecOCMaxTransmitRetries` times.⌋

[SWS_SecOC_00077]

Upstream requirements: SRS_BSW_00385

⌈If the lower layer transport protocol module reports an error during transmission of a Secured I-PDU using the return value `E_NOT_OK`, the SecOC module shall not perform any error handling other than skipping the confirmation of the transmission request for the corresponding Authentic I-PDU to the upper layer module.⌋

[SWS_SecOC_00151]

Upstream requirements: SRS_SecOC_00021, SRS_BSW_00385

⌈If the CSM module reports a recoverable error (example: `E_BUSY`, `QUEUE_FULL`) during authentication of an Authentic I-PDU , the SecOC module shall not provide a Secured I-PDU to the lower layer. It shall keep that Authentic I-PDU (if not overwritten by an incoming Authentic I-PDU of the same type) to start the authentication with the next call of the scheduled main function until the number of additional authentication attempts for that Authentic I-PDU has reached its limits.⌋

[SWS_SecOC_00155]

Upstream requirements: SRS_BSW_00385

⌈If the number of attempts for an Authentic I-PDU has reached the limit `SecOCAuthenticationBuildAttempts` that defines the maximum number of freshness values provided by the freshness manager, the SecOC module shall report `SECOC_E_CRYPTO_FAILURE` to the DET module.⌋

[SWS_SecOC_00108]

Upstream requirements: SRS_BSW_00385

⌈If the SecOC module is not able to serve any upper layer or lower layer request during transmission of an Authentic I-PDU due to an arbitrary internal error, it shall return this request with `E_NOT_OK`.⌋

[SWS_SecOC_00217]

Upstream requirements: SRS_SecOC_00021

⌈If the upper layer module requests a cancelation of an ongoing reception of the Authentic I-PDU by calling `SecOC_TpCancelReceive`, the SecOC module shall immediately inform the lower layer transport protocol module to cancel the ongoing reception of the Secured I-PDU, stop all internal actions related to the Authentic I-PDU, and free all related buffers.⌋

[SWS_SecOC_00260]

Upstream requirements: SRS_BSW_00385

⌈If the upper layer transport protocol module reports `BUFREQ_E_BUSY` in a call to `PduR_SecOCTpCopyTxData` then SecOC shall retry the call in the next subsequent call of its scheduled main function.⌋

[SWS_SecOC_00266]

Upstream requirements: SRS_BSW_00385

⌈If the upper layer transport protocol module reports `BUFREQ_E_NOT_OK` in a call to `PduR_SecOCTpCopyTxData` then SecOC shall immediately abort the transmission via calling `PduR_SecOCTpTxConfirmation` with `E_NOT_OK` result, shall stop all internal actions related to the Authentic I-PDU, and shall free all related buffers.⌋

7.5 Verification of incoming PDUs

The term verification describes the process of comparing the Authentication Information contained in a Secured I-PDU with the Authentication Information calculated on basis of the local Data Identifier, the local Freshness Value and the Authentic I-PDU contained in the Secured I-PDU.

The process of verifying incoming Secured I-PDUs is described in general terms in Section 7.1.3. This section refines the general description with respect to requirements arising from the integration with the PduR module considering different bus interfaces and transport protocols. The overall interaction with the PduR module and the verification of Secured I-PDUs is organized as described in the following scheme:

1. For each indication of an incoming Secured I-PDU from a lower layer bus interface or transport protocol module, the SecOC module takes the role of an upper layer communication module and thus serves all lower layer requests that are necessary to receive the complete Secured I-PDU.
2. The SecOC module copies the Secured I-PDU into its own memory.
3. Thereafter, when the complete Secured I-PDU is available and during the next scheduled call of its main function, the SecOC module verifies the contents of the Secured I-PDU according to [SWS_SecOC_00040].
4. If the verification fails and the parameter `SecOCIgnoreVerificationResult` is configured to FALSE, the SecOC module drops the Secured I-PDU.
5. If the verification succeeds or the verification fails and the parameter `SecOCIgnoreVerificationResult` is configured to TRUE, the SecOC module takes the role of a lower layer communication module and calls `PduR_SecOC[If|Tp]RxIndication` for the Authentic I-PDU.
6. The SecOC reports the verification results according to [SWS_SecOC_00048].

Thus, SecOC decouples the interaction between upper layer modules and lower layer modules. The SecOC module manages the interaction with lower layer module until it has copied the complete Secured I-PDU into its own buffer. It does so without affecting the upper layer module. Thereafter, it verifies the contents of the Secured I-PDU and, dependent on the verification results, initiates the transmission of the Authentic I-PDU to the upper layer communication module.

[SWS_SecOC_00214]

Upstream requirements: SRS_SecOC_00021, SRS_SecOC_00022

⌈In case the `SecOCReceptionOverflowStrategy` is set to `REPLACE`, the SecOC module shall free all buffer related to a Secured I-PDU if the reception of a Secured I-PDU with the same Pdu Identifier has been initiated.⌋

[SWS_SecOC_00215]

Upstream requirements: SRS_SecOC_00021, SRS_SecOC_00022

⌈In case the `SecOCReceptionOverflowStrategy` is set to `REJECT` and SecOC is currently busy with the same Secured I-PDU, the SecOC module shall ignore any subsequent call of `SecOC_RxIndication` and return `BUFREQ_E_NOT_OK` for any subsequent call of `SecOC_StartOfReception`.⌋

[SWS_SecOC_00204]

Upstream requirements: SRS_SecOC_00026

⌈SecOC shall provide separate buffers for the incoming Secured I-PDU, Cryptographic I-PDU and the resulting Authentic I-PDU.⌋

Note: Thus, lower layer updates of Secured I-PDUs could be processed without affecting ongoing deliveries of an Authentic I-PDU to the upper layer communication modules.

[SWS_SecOC_00216]

Upstream requirements: SRS_SecOC_00021, SRS_SecOC_00022

⌈In case the SecOCReceptionOverflowStrategy is set to QUEUE and SecOC is currently busy with the same Secured I-PDU, the SecOC module shall additionally receive the Secured I-PDU and queue them for a subsequent processing after the currently processed Secured I-PDU is finalized. In case the limit which is given by SecOCReceptionQueueSize is reach any further reception shall be rejected.⌋

[SWS_SecOC_00205]

Upstream requirements: SRS_SecOC_00026

⌈For each Secured I-PDU having SecOCRxSecuredPduCollection present in the corresponding SecOCRxSecuredPduLayer SecOC shall buffer only the last Authentic I-PDU and Cryptographic I-PDU it has received. If a buffer has already been filled with a previous I-PDU, the previous I-PDU is overwritten.⌋

Note: An Authentic I-PDU and its corresponding Cryptographic I-PDU must be received in direct succession but their order does not matter. This can be realized for example via priority handling dependent on the underlying bus system.

[SWS_SecOC_00206]

Upstream requirements: SRS_SecOC_00026

⌈SecOC shall construct and the Secured I-PDU immediately after it has received both the respective Authentic I-PDU and Cryptographic I-PDU. If SecOC_VerifyStatusOverride is used, the verification result and I-PDU are handled according to overrideStatus value.⌋

[SWS_SecOC_00207]

Upstream requirements: SRS_SecOC_00026

⌈If the subsequent verification of the resulting Secured I-PDU is successful, then SecOC shall clear the buffers of both the Authentic and Cryptographic I-PDU.⌋

[SWS_SecOC_00257]

Upstream requirements: SRS_SecOC_00010, SRS_SecOC_00012, SRS_SecOC_00013

⌈For a Secured Rx I-PDU with SecOCAuthPduHeaderLength = 0 or not configured and SecOCDynamicRuntimeLengthHandling set to FALSE, the SecOC module shall extract the Authentic I-PDU by using the configured length of the corresponding global PDU.⌋

[SWS_SecOC_00258]

Upstream requirements: SRS_SecOC_00010, SRS_SecOC_00012, SRS_SecOC_00013

⌈For a Secured Rx I-PDU with `SecOCAuthPduHeaderLength` = 0 or not configured and `SecOCDynamicRuntimeLengthHandling` set to TRUE, the SecOC module shall extract the Authentic I-PDU by using the length provided by the lower layer.⌋

[SWS_SecOC_00259]

Upstream requirements: SRS_SecOC_00010, SRS_SecOC_00012, SRS_SecOC_00013

⌈For a Secured Rx I-PDU with `SecOCAuthPduHeaderLength` > 0, the SecOC module shall extract the Authentic I-PDU using the length provided at runtime within the Secured I-PDU Header.⌋

7.5.1 Verification during bus interface reception

When a Secured I-PDU is received by means of a lower layer bus interface(e.g. CanIf, FrIf), the PduR module calls `SecOC_RxIndication` to inform the SecOC module for each incoming Secured I-PDU. During the processing of `SecOC_RxIndication`, the SecOC module copies the Authentic I-PDU to its own buffer.

[SWS_SecOC_00268]

Upstream requirements: SRS_SecOC_00010, SRS_SecOC_00012, SRS_SecOC_00013

⌈During reception of a static length (Secured / Authentic / Cryptographic) I-PDU, i.e. `SecOCDynamicRuntimeLengthHandling` set to FALSE, by means of a lower layer bus interface and when `SecOC_RxIndication` has been called, the SecOC module shall silently discard this I-PDU in case of the received length is smaller than the configured length.⌋

Note: Static PDUs will normally be sent with configured length, therefore a mismatch between received length and configured length is seen as an error scenario. Also as static PDUs do not contain header length Information it could lead to errors in case of a shorter length in combination with padding bytes.

[SWS_SecOC_00078]

Upstream requirements: SRS_SecOC_00010, SRS_SecOC_00012, SRS_SecOC_00013

⌈During reception of an (Secured / Authentic / Cryptographic) I-PDU by means of a lower layer bus interface and when `SecOC_RxIndication` has been called, the SecOC module shall copy the I-PDU into the according buffer according to the minimum of received length and configured length of this I-PDU. The copied length shall then be used for all further reception processings.⌋

Note: Copying only minimum of configured and passed length ensures that buffer cannot be overwritten and that non-expected data (which was maybe added due to padding) is discarded. For reception from TP this restriction is not needed as TP ensures a valid length value passed. For dynamic length PDUs with a shorter length than configured only the length provided will be copied. Also for dynamic length PDUs [TPS_SYST_02189] ensures that a reliable length information is available.

Thereafter, the actual verification of anincoming Secured I-PDU is initiated during the next call of the scheduled main function. The SecOC module extracts the Authentic I-PDU, the Authentication Information from the Secured I-PDU. The SecOC module verifies the authenticity and freshness of the Authentic I-PDU according to [SWS_SecOC_00040]. If the verification is successful, the SecOC indicates the reception of the Authentic I-PDU by calling `PduR_SecOC[If|Tp]RxIndication` for the Authentic I-PDU. If the verification fails, the SecOC drops the PDU and does not call `PduR_SecOC[If|Tp]RxIndication`.

[SWS_SecOC_00079]

Upstream requirements: SRS_SecOC_00010, SRS_SecOC_00012, SRS_SecOC_00013

⌈During reception of a Secured I-PDU that is received by means of a lower layer bus interface, the SecOC module shall verify the Authenticator according to the overall approach specified in [SWS_SecOC_00040]. The verification shall be processed in the scheduled main function.⌋

[SWS_SecOC_00080]

Upstream requirements: SRS_SecOC_00010, SRS_SecOC_00012, SRS_SecOC_00013

⌈During reception of a Secured I-PDU that is received by means of a lower layer bus interface and if the verification eventually succeeds, the SecOC module shall call `PduR_SecOC[If|Tp]RxIndication` referencing the Authentic I-PDU that is contained in the Secured I-PDU.⌋

[SWS_SecOC_00081]

Upstream requirements: SRS_SecOC_00010, SRS_SecOC_00012, SRS_SecOC_00013

⌈During reception of a Secured I-PDU that is received by means of a lower layer bus interface and if the verification fails and the `SecOCIgnoreVerificationResult` is configured to TRUE, the SecOC module shall call `PduR_SecOC[If|Tp]RxIndication` referencing the Authentic I-PDU that is contained in the Secured I-PDU.⌋

Note: If the verification eventually fails, the SecOC module does not call `PduR_SecOC[If|Tp]RxIndication` for the Authentic I-PDU that is contained in the Secured I-PDU.

7.5.2 Verification during transport protocol reception

When a Secured I-PDU is received by means of a lower layer transport protocol interface (e.g. CanTp, FrTp), the PduR module calls SecOC_StartOfReception to notify the SecOC module that the reception process of the respective Secured I-PDU will start.

[SWS_SecOC_00082]

Upstream requirements: SRS_SecOC_00010, SRS_SecOC_00012, SRS_SecOC_00013

⌈During reception of a Secured I-PDU that is received by means of a lower layer transport protocol interface and when SecOC_StartOfReception is called, the SecOC module shall provide buffer capacities to store the complete Secured I-PDU. Further it shall forward the SecOC_StartOfReception call by calling PduR_SecOCTpStartOfReception in case SecOCPduType is configured to SECOC_TPPDU.⌋

Note: In case the upper layer does not accept the reception, SecOC should not accept the reception as well.

When the lower layer iteratively indicates the reception of the individual NPDUs that constitute the Secured I-PDU (i.e. when SecOC_CopyRxData is called), the SecOC module copies the NPDUs to its own buffer.

[SWS_SecOC_00083]

Upstream requirements: SRS_SecOC_00010, SRS_SecOC_00012, SRS_SecOC_00013

⌈During reception of a Secured I-PDU that is received by means of a lower layer transport protocol interface and when SecOC_CopyRxData is called, the SecOC module shall copy the NPDUs addressed by SecOC_CopyRxData into its own buffers. Finally, it returns from SecOC_CopyRxData.⌋

Finally, when the lower layer confirms the complete reception of the Secured I-PDU via SecOC_TpRxIndication and thus the complete Secured I-PDU is available in the buffer of the SecOC module for further processing, the SecOC module starts the verification of the Authentication Information according to Section 7.1.3 during its next scheduled call of its main function.

[SWS_SecOC_00084]

Upstream requirements: SRS_SecOC_00010, SRS_SecOC_00012, SRS_SecOC_00013

⌈During reception of a Secured I-PDU that is received by means of a lower layer transport protocol interface and when SecOC_TpRxIndication is called, the SecOC module shall returns SecOC_TpRxIndication without any further processing.⌋

[SWS_SecOC_00085]

Upstream requirements: SRS_SecOC_00010, SRS_SecOC_00012, SRS_SecOC_00013

⌈During reception of a Secured I-PDU that is received by means of a lower layer transport protocol interface and when `SecOC_TpRxIndication` has been called, the SecOC module shall verify the contents of the Secured I-PDU according to the process described in [SWS_SecOC_00040].⌋

[SWS_SecOC_00086]

Upstream requirements: SRS_SecOC_00010, SRS_SecOC_00012, SRS_SecOC_00013

⌈During reception of a Secured I-PDU that is received by means of a lower layer transport protocol interface and when the verification eventually succeeds, the SecOC module shall call `PduR_SecOCIfRxIndication` with references to the Authentic I-PDU contained in the Secured I-PDU in case `SecOCPduType` is configured to `SECOC_IFPDU`.

In case `SecOCPduType` is configured to `SECOC_TPPDU` SecOC shall forward in advance all data to the upper layer by first calling `PduR_SecOCTpCopyRxData` and afterwards `PduR_SecOCTpRxIndication` with references to the Authentic I-PDU contained in the Secured I-PDU.⌋

[SWS_SecOC_00088]

Upstream requirements: SRS_SecOC_00010, SRS_SecOC_00012, SRS_SecOC_00013

⌈During reception of a Secured I-PDU that is received by means of a lower layer transport protocol interface and when the verification fails and the `SecOCIgnoreVerificationResult` is configured to TRUE, the SecOC module shall call `PduR_SecOCIfRxIndication` with references to the Authentic I-PDU contained in the Secured I-PDU in case `SecOCPduType` is configured to `SECOC_IFPDU`.

In case `SecOCPduType` is configured to `SECOC_TPPDU` SecOC shall forward in advance all data to the upper layer by first calling `PduR_SecOCTpCopyRxData` and afterwards `PduR_SecOCTpRxIndication` with references to the Authentic I-PDU contained in the Secured I-PDU.⌋

[SWS_SecOC_00213]

Upstream requirements: SRS_BSW_00385

⌈In case the SecOC frees buffers related to a Secured I-PDU (see [SWS_SecOC_00087]) and `SecOCPduType` is configured to `SECOC_TPPDU` the SecOC shall cancel the reception in the upper layer (negative `PduR_SecOCTpRxIndication`).⌋

[SWS_SecOC_00087]

Upstream requirements: SRS_SecOC_00021, SRS_SecOC_00022

⌈The SecOC module shall free all buffer related to a Secured I-PDU either if

1. it has passed the respective authenticated I-PDU to the PduR via `PduR_SecOCIfRxIndication` or `PduR_SecOCTpRxIndication`,
2. the verification of a Secured I-PDU eventually failed,
3. the transmission of a Secured I-PDU has been canceled by the upper or lower layer.

⌋

[SWS_SecOC_00255]

Upstream requirements: SRS_SecOC_00032

⌈The SecOC module shall receive the complete Secured I-PDU in its internal memory before starting any copying of the corresponding Authentic I-PDU.⌋

7.5.3 Skipping Authentication for Secured I-PDUs at SecOC

[SWS_SecOC_00265]

Upstream requirements: SRS_BSW_00385

⌈For a Rx Secured I-PDU with `SecOCSecuredRxPduVerification`=false, the SecOC module shall extract the Authentic I-PDU without Authentication.⌋

7.5.4 Error handling and discarding of reception

[SWS_SecOC_00089]

Upstream requirements: SRS_BSW_00385

⌈If the lower layer transport protocol module reports an error by returning something else than `E_OK` during reception of a Secured I-PDU using `SecOC_TpRxIndication`, the SecOC module shall drop the Secured I-PDU and free all corresponding buffers.⌋

[SWS_SecOC_00121]

Upstream requirements: SRS_SecOC_00022, SRS_BSW_00385

⌈If the CSM module reports an error during verification (verification attempt returns `E_NOT_OK`) of a Secured I-PDU, the SecOC module shall not provide the Authentic I-PDU. It shall keep the Secured I-PDU (if not overwritten by an incoming Secured I-PDU of the same type) and start the verification with the next call of the scheduled main function.⌋

[SWS_SecOC_00208]

Upstream requirements: SRS_SecOC_00026

⌈If SecOC has received both an Authentic I-PDU and a Cryptographic PDU and the verification of the resulting Secured I-PDU fails, both the Authentic and Cryptographic I-PDU shall remain buffered and verification shall be reattempted each time new data for any of them is received.⌋

Note: This and the above requirement ensure that even if either an Authentic I-PDU or a Cryptographic I-PDU is lost in transit, SecOC will still function as expected as soon as an Authentic I-PDU and its corresponding Cryptographic I-PDU are received in direct succession.

[SWS_SecOC_00109]

Upstream requirements: SRS_BSW_00385

⌈If the SecOC module is not able to serve any upper layer or lower layer request during reception of A Secured I-PDU due to an arbitrary internal error, it shall return this request with `E_NOT_OK`.⌋

[SWS_SecOC_00263]

Upstream requirements: SRS_BSW_00385

⌈For a Rx Secured I-PDU with `SecOCAuthPduHeaderLength` > 0 and the length of Authentic I-PDU in the Header is longer than configured length (in case of dynamic length IPdus (containing a dynamical length signal), this value indicates the maximum data length) of the Authentic I-PDU, the SecOC module shall discard the I-PDU. In such case with `SecOC_StartOfReception`, `BUFREQ_E_NOT_OK` shall be returned (see [SWS_COMTYPE_00012]).

Note: `SecOC_RxIndication` has no return value.⌋

[SWS_SecOC_00264]

Upstream requirements: SRS_BSW_00385

⌈For a Rx Secured I-PDU with `SecOCAuthPduHeaderLength` > 0, the SecOC module shall process Secured I-PDU Header, Authentic I-PDU (with the length specified by the Header), Freshness Value and Authenticator of the Rx Secured I-PDU. The rest of bytes in the Secured I-PDU shall be discarded.⌋

Note: In case of static PDUs (e.g. if `SecOCDynamicRuntimeLengthHandling` set to FALSE) having no header part for secured I-PDU and originating from a bus which does add padding (CANFD and Flexray), the configured SduLength should be taken to determine Freshness / MAC position.

[SWS_SecOC_00267]

Upstream requirements: SRS_SecOC_00010, SRS_SecOC_00012

⌈If the upper layer transport protocol module reports `BUFREQ_E_NOT_OK` in a call to `PduR_SecOCTpCopyRxData` then SecOC shall immediately abort the reception via calling `PduR_SecOCTpRxIndication` with `E_NOT_OK` result, shall stop all internal actions related to the Secured I-PDU, and shall free all related buffers.⌋

7.6 Gateway functionality

The SecOC module supports authentication and verification for I-PDUs that are routed from one source bus to one or more destination busses. This allows for the realization of re-authentication gateways that can be used to realize networks with different security zones or properties. The actions necessary to support the required gateway functionality can be simply derived from the authentication and verification scenarios in Sections 7.4 and 7.5. Each authentication or verification process for a given I-PDU need to be configured separately. This functionality includes:

- authentication of outgoing I-PDUs,
- verification of incoming I-PDUs,
- re-authentication gateways, i.e. the verification of incoming I-PDUs in combination of their immediate re-authentication, when the I-PDU is routed to another lower layer module.

Note: "Gatewaying-on-the-fly" is not supported by SecOC

7.7 Multicore Distribution

In order to provide a load distribution amongst different partitions, the different parts of the Crypto-Stack shall be allocated to the different partitions. Hereby it shall be supported that such a partitioning happens on a crypto instance basis, i.e., the crypto driver instances shall be locatable onto different distinct partitions.

In order to support such a flexible allocation the main threads of execution in the SecOC module (namely the respective MainFunctions) can be split into different MainFunctions (at least one per partition). This way the flow through the crypto stack stays within the scope of a single partition and therefore does not require special multi-partition capable means.

The inter-partition communication between SecOC and PduR is managed by PduR.

In order to manage different timing requirements each MainFunction instance defines its time base individually.

[SWS_SecOC_00276]

Upstream requirements: SRS_BSW_00432

⌈`SecOCTxPduProcessing`s shall be processed within the MainFunction, which is referenced via `SecOCTxPduMainFunctionRef` (see [ECUC_SecOC_00111]).⌋

7.8 Security Events

[SWS_SecOC_00273]

Upstream requirements: RS_Ids_00810

⌈If security event reporting has been enabled for the SecOC module (`SecOCEnableSecurityEventReporting` = true) the respective security events shall be reported to the IdsM via the interfaces defined in [4].⌋

[SWS_SecOC_00274]

Upstream requirements: SRS_BSW_00432

⌈The table [SWS_SecOC_00115] lists the security events which are standardized for the SecOC together with their trigger conditions.⌋

[SWS_SecOC_00115] Security events for SecOc

Status: DRAFT
Upstream requirements: RS_Ids_00810

⌈

Name	*Description*	*ID*
SEV_SECOC_MAC_VERIFICATION_FAILED	MAC verification of a received PDU failed.	44
SEV_SECOC_FRESHNESS_NOT_AVAILABLE	Failed to get freshness value from FvM.	45

⌋

[SWS_SecOC_92000] Security event context data definition: SEV_SECOC_MAC_VERIFICATION_FAILED

Status: DRAFT
Upstream requirements: RS_Ids_00810

⌈

SEV Name	*SEV_SECOC_MAC_VERIFICATION_FAILED*	
ID	44	
Description	MAC verification of a received PDU failed.	
Context Data Version	1	
Context Data	*Data Type*	*Allowed Values*
DataId	uint16	

⌋

7.9 Error Classification

Section "Error Handling" of the document [4] "General Specification of Basic Software Modules" describes the error handling of the Basic Software in detail. Above all, it constitutes a classification scheme consisting of five 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

[SWS_SecOC_00101] Definiton of development errors in module SecOC

Upstream requirements: SRS_BSW_00337, SRS_BSW_00385, SRS_BSW_00386

⌈

Type of error	*Related error code*	*Error value*
An API service was called with a NULL pointer	SECOC_E_PARAM_POINTER	0x01
API service used without module initialization	SECOC_E_UNINIT	0x02
Invalid I-PDU identifier	SECOC_E_INVALID_PDU_SDU_ID	0x03
Crypto service failed	SECOC_E_CRYPTO_FAILURE	0x04
initialization of SecOC failed	SECOC_E_INIT_FAILED	0x07

⌋

7.9.2 Runtime Errors

[SWS_SecOC_00114] Definiton of runtime errors in module SecOC

Upstream requirements: SRS_BSW_00337, SRS_BSW_00385, SRS_BSW_00386

⌈

Type of error	*Related error code*	*Error value*
NO freshness value available from the Freshness Manager	SECOC_E_FRESHNESS_FAILURE	0x08

⌋

7.9.3 Production Errors

There are no production errors.

7.9.4 Extended Production Errors

There are no extended production errors.

7.10 Security Profiles

7.10.1 Secured area within a Pdu

[SWS_SecOC_00311]

Upstream requirements: SRS_SecOC_00003

⌈If the parameter SecOCSecuredTxPduOffset or SecOCSecuredRxPduOffset is available, the applied Security Profile shall only consider the bytes starting with the configured offset.⌋

[SWS_SecOC_00312]

Upstream requirements: SRS_SecOC_00003

⌈If the parameter SecOCSecuredTxPduLength or SecOCSecuredRxPduLength is available, the applied Security Profile shall only consider the configured length.⌋

[SWS_SecOC_00313]

Upstream requirements: SRS_SecOC_00003

⌈If the sum of configured value of SecOCSecuredTxPduLength and SecOCSecuredTxPduOffset is longer than the PduInfoPtr->SduLength provided to SecOC_IfTransmit or SecOC_TpTransmit, this Pdu shall be discarded and E_NOT_OK shall be returned.⌋

[SWS_SecOC_00314]

Upstream requirements: SRS_SecOC_00003

⌈If the sum of configured value of SecOCSecuredRxPduLength and SecOCSecuredRxPduOffset are longer than the received Pdu length itself, this Pdu shall be discarded.⌋

7.10.2 Overview of security profiles

The specification of the module Secure Onboard Communication allows different configurations for which cryptographic algorithms and modes to use for the MAC calculation and how the truncation of the MAC and freshness value (if applicable) shall be done. The security profiles provide a consistent set of values for a subset of configuration parameters that are relevant for the configuration of Secure Onboard Communication.

[SWS_SecOC_00190]

Upstream requirements: SRS_SecOC_00003

⌈Each Security Profile shall provide the configuration values for the authentication algorithm (parameter `algorithmFamily`, `algorithmMode` and `algorithmSecondaryFamily` in `CryptoServicePrimitive`), length of freshness Value, if applicable (parameter `SecOCFreshnessValueLength`), length of truncated Freshness Value (parameter `SecOCFreshnessValueTruncLength`), length of truncated MAC (parameter `SecOCAuthInfoTruncLength`), and a description of the profile.⌋

[SWS_SecOC_00191]

Upstream requirements: SRS_SecOC_00003

⌈A security profile shall be defined by the following mandatory parameters in the System Template:

+ `algorithmFamily`:String [0..1]

+ `algorithmMode` :String [0..1]

+ `algorithmSecondaryFamily` :String [0..1]

+ `authInfoTxLength` :PositiveInteger

+ `freshnessValueLength` :PositiveInteger

+ `freshnessValueTxLength` :PositiveInteger⌋

7.10.3 SecOC Profile 1 (or 24Bit-CMAC-8Bit-FV)

[SWS_SecOC_00192]

Upstream requirements: SRS_SecOC_00003

⌈Using the CMAC algorithm based on AES-128 according to NIST SP 800-38B to calculate the MAC, use the eight least significant bit of the freshness value as truncated freshness value and use the 24 most significant bits of the MAC as truncated MAC.⌋

Parameter	Configuration value
The algorithm for the MAC (parameter algorithmFamily)	CRYPTO_ALGOFAM_AES
The algorithm mode for the MAC (parameter algorithmMode)	CRYPTO_ALGOMODE_CMAC
Additional algorithm family configuration (parameter algorithmSecondaryFamily, not used in this profile)	CRYPTO_ALGOFAM_NOT_SET
Length of Freshness Value (parameter SecOCFreshnessValueLength)	Not Specified
length of truncated Freshness Value (parameter SecOCFreshnessValueTruncLength	8 bits
length of truncated MAC (parameter SecOCAuthInfoTruncLength)	24 bits

7.10.4 SecOC Profile 2 (or 24Bit-CMAC-No-FV)

[SWS_SecOC_00193]

Upstream requirements: SRS_SecOC_00003

⌈Using the CMAC algorithm based on AES-128 according to NIST SP 800-38B to calculate the MAC, don't use any freshness value at all and use the 24 most significant bits of the MAC as truncated MAC.

The profile shall only be used if no synchronized freshness value is established. There is no restriction to a special bus.⌋

Parameter	Configuration value
The algorithm for the MAC (parameter algorithmFamily)	CRYPTO_ALGOFAM_AES
The algorithm mode for the MAC (parameter algorithmMode)	CRYPTO_ALGOMODE_CMAC
Additional algorithm family configuration (parameter algorithmSecondaryFamily, not used in this profile)	CRYPTO_ALGOFAM_NOT_SET
Length of Freshness Value (parameter SecOCFreshnessValueLength)SecOC	0
length of truncated Freshness Value (parameter SecOCFreshnessValueTruncLength	0 bits
length of truncated MAC (parameter SecOCAuthInfoTruncLength)	24 bits

7.10.5 SecOC Profile 3 (or JASPAR)

[SWS_SecOC_00194]

Upstream requirements: SRS_SecOC_00003

⌈This profile depicts one configuration and usage of the JasPar counter base FV with Master-Slave Synchronization method.

It uses the CMAC algorithm based on AES-128 according to NIST SP 800-38B Appendix-A to calculate the MAC. Use the 4 least significant bits of the freshness value as truncated freshness value and use the 28 most significant bits of the MAC as truncated MAC.

Freshness Value provided to SecOC shall be constructed as described in the [UC_SecOC_00202]. The profile shall be used for CAN.⌋

Parameter	Configuration value
The algorithm for the MAC (parameter algorithmFamily)	CRYPTO_ALGOFAM_AES
The algorithm mode for the MAC (parameter algorithmMode)	CRYPTO_ALGOMODE_CMAC
Additional algorithm family configuration (parameter algorithmSecondaryFamily, not used in this profile)	CRYPTO_ALGOFAM_NOT_SET
Length of Freshness Value (parameter SecOCFreshnessValueLength)	64 bits
length of truncated Freshness Value (parameter SecOCFreshnessValueTruncLength	4 bits
length of truncated MAC (parameter SecOCAuthInfoTruncLength)	28 bits

8 API specification

8.1 Imported types

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

[SWS_SecOC_00103] Definition of imported datatypes of module SecOC

Upstream requirements: SRS_BSW_00301

⌈

Module	*Header File*	*Imported Type*
Comtype	ComStack_Types.h	BufReq_ReturnType
	ComStack_Types.h	PduIdType
	ComStack_Types.h	PduInfoType
	ComStack_Types.h	PduLengthType
	ComStack_Types.h	RetryInfoType
	ComStack_Types.h	TpDataStateType
Csm	Rte_Csm_Type.h	Crypto_OperationModeType
	Rte_Csm_Type.h	Crypto_VerifyResultType
IdsM	IdsM_Types.h	IdsM_SecurityEventIdType
Std	Std_Types.h	Std_ReturnType
	Std_Types.h	Std_VersionInfoType

⌋

8.2 Type definitions

8.2.1 SecOC_ConfigType

[SWS_SecOC_00104] Definition of datatype SecOC_ConfigType

Upstream requirements: SRS_SecOC_00001, SRS_SecOC_00003

⌈

Name	SecOC_ConfigType	
Kind	Structure	
Elements	implementation specific	
	Type	–
	Comment	The content of the configuration data structure is implementation specific.
Description	Configuration data structure of SecOC module	

▽

△

Available via	SecOC.h

⌋

8.2.2 SecOC_StateType

[SWS_SecOC_00162] Definition of datatype SecOC_StateType

Upstream requirements: SRS_SecOC_00005

⌈

Name	SecOC_StateType		
Kind	Enumeration		
Range	SECOC_UNINIT	–	SecOC module is not initialized
	SECOC_INIT	–	SecOC module is initialized
Description	States of the SecOC module		
Available via	SecOC.h		

⌋

8.3 Function definitions

8.3.1 SecOC_Init

[SWS_SecOC_00106] Definition of API function SecOC_Init

Upstream requirements: SRS_BSW_00101, SRS_BSW_00323, SRS_BSW_00358, SRS_BSW_-00359, SRS_BSW_00414, SRS_SecOC_00006

⌈

Service Name	SecOC_Init	
Syntax	`void SecOC_Init ( const SecOC_ConfigType* config )`	
Service ID [hex]	0x01	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	config	Pointer to a selected configuration structure
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	

▽

△

Description	Initializes the the SecOC module. Successful initialization leads to state SecOC_INIT. In configurations, in which SecOC is assigned to more than one partition (i.e. SecOC_Main Functions are mapped to partitions), SecOC may provide one init function per partition. The decision on whether a single SecOC_Init() function or one per partition is provided is implementation-specific. In case a given implementation provides one SecOC_Init() function per partition, it is up to the implementation to devise a naming pattern that prevents name clashes among the different SecOC_Init() functions (e.g., by adding a suffix containing short name the EcucPartition.
Available via	SecOC.h

⌋

8.3.2 SecOC_DeInit

[SWS_SecOC_00161] Definition of API function SecOC_DeInit

Upstream requirements: SRS_BSW_00323, SRS_BSW_00359, SRS_SecOC_00006, SRS_SecOC_00020

⌈

Service Name	SecOC_DeInit
Syntax	`void SecOC_DeInit ( void )`
Service ID [hex]	0x05
Sync/Async	Synchronous
Reentrancy	Non Reentrant
Parameters (in)	None
Parameters (inout)	None
Parameters (out)	None
Return value	None
Description	This service stops the secure onboard communication. All buffered I-PDU are removed and have to be obtained again, if needed, after SecOC_Init has been called. By a call to SecOC_De Init the AUTOSAR SecOC module is put into a not initialized state (SecOC_UNINIT).
Available via	SecOC.h

⌋

[SWS_SecOC_00157]

Upstream requirements: SRS_BSW_00323, SRS_SecOC_00006

⌈Within `SecOC_DeInit` the module shall clear all internal global variables and the buffers of the SecOC I-PDUs.⌋

8.3.3 SecOC_GetVersionInfo

[SWS_SecOC_00107] Definition of API function SecOC_GetVersionInfo

Upstream requirements: SRS_BSW_00323, SRS_BSW_00359, SRS_BSW_00407, SRS_BSW_-00369, SRS_BSW_00003, SRS_BSW_00402

⌈

Service Name	SecOC_GetVersionInfo	
Syntax	`void SecOC_GetVersionInfo ( Std_VersionInfoType* versioninfo )`	
Service ID [hex]	0x02	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	None	
Parameters (inout)	None	
Parameters (out)	versioninfo	Pointer to where to store the version information of this module.
Return value	None	
Description	Returns the version information of this module.	
Available via	SecOC.h	

⌋

8.3.4 SecOC_IfTransmit

[SWS_SecOC_00112] Definition of API function SecOC_IfTransmit

Upstream requirements: SRS_BSW_00323, SRS_BSW_00357, SRS_BSW_00369, SRS_BSW_-00449

⌈

Service Name	SecOC_IfTransmit	
Syntax	`Std_ReturnType SecOC_IfTransmit ( PduIdType TxPduId, const PduInfoType* PduInfoPtr )`	
Service ID [hex]	0x49	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different PduIds. Non reentrant for the same PduId.	
Parameters (in)	TxPduId	Identifier of the PDU to be transmitted
	PduInfoPtr	Length of and pointer to the PDU data and pointer to MetaData.
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: Transmit request has been accepted. `E_NOT_OK`: Transmit request has not been accepted.

▽

△

Description	Requests transmission of a PDU.
Available via	SecOC.h

⌋

For detailed description, see Section 7.4.

8.3.5 SecOC_TpTransmit

[SWS_SecOC_91008] Definition of API function SecOC_TpTransmit

Upstream requirements: SRS_BSW_00323, SRS_BSW_00357, SRS_BSW_00369, SRS_BSW_-00449

⌈

Service Name	SecOC_TpTransmit	
Syntax	`Std_ReturnType SecOC_TpTransmit ( PduIdType TxPduId, const PduInfoType* PduInfoPtr )`	
Service ID [hex]	0x53	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different PduIds. Non reentrant for the same PduId.	
Parameters (in)	TxPduId	Identifier of the PDU to be transmitted
	PduInfoPtr	Length of and pointer to the PDU data and pointer to MetaData.
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: Transmit request has been accepted. `E_NOT_OK`: Transmit request has not been accepted.
Description	Requests transmission of a PDU.	
Available via	SecOC.h	

⌋

For detailed description, see Section 7.4.

8.3.6 SecOC_IfCancelTransmit

[SWS_SecOC_00113] Definition of API function SecOC_IfCancelTransmit

Upstream requirements: SRS_BSW_00323, SRS_BSW_00357, SRS_BSW_00449, SRS_SecOC_00012

⌈

Service Name	SecOC_IfCancelTransmit	
Syntax	`Std_ReturnType SecOC_IfCancelTransmit ( PduIdType TxPduId )`	
Service ID [hex]	0x4a	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different PduIds. Non reentrant for the same PduId.	
Parameters (in)	TxPduId	Identification of the PDU to be cancelled.
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: Cancellation was executed successfully by the destination module. `E_NOT_OK`: Cancellation was rejected by the destination module.
Description	Requests cancellation of an ongoing transmission of a PDU in a lower layer communication module.	
Available via	SecOC.h	

⌋

8.3.7 SecOC_TpCancelTransmit

[SWS_SecOC_91009] Definition of API function SecOC_TpCancelTransmit

Upstream requirements: SRS_BSW_00323, SRS_BSW_00357, SRS_BSW_00449, SRS_SecOC_00012

⌈

Service Name	SecOC_TpCancelTransmit	
Syntax	`Std_ReturnType SecOC_TpCancelTransmit ( PduIdType TxPduId )`	
Service ID [hex]	0x54	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different PduIds. Non reentrant for the same PduId.	
Parameters (in)	TxPduId	Identification of the PDU to be cancelled.
Parameters (inout)	None	
Parameters (out)	None	

▽

△

Return value	Std_ReturnType	E_OK: Cancellation was executed successfully by the destination module. E_NOT_OK: Cancellation was rejected by the destination module.
Description	Requests cancellation of an ongoing transmission of a PDU in a lower layer communication module.	
Available via	SecOC.h	

⌋

8.3.8 SecOC_TpCancelReceive

[SWS_SecOC_91010] Definition of API function SecOC_TpCancelReceive

Upstream requirements: SRS_SecOC_00003, SRS_SecOC_00012

⌈

Service Name	SecOC_TpCancelReceive	
Syntax	`Std_ReturnType SecOC_TpCancelReceive ( PduIdType RxPduId )`	
Service ID [hex]	0x4c	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	RxPduId	Identification of the PDU to be cancelled.
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: Cancellation was executed successfully by the destination module. E_NOT_OK: Cancellation was rejected by the destination module.
Description	Requests cancellation of an ongoing reception of a PDU in a lower layer transport protocol module.	
Available via	SecOC.h	

⌋

8.3.9 SecOC_VerifyStatusOverride

[SWS_SecOC_00122] Definition of API function SecOC_VerifyStatusOverride

Upstream requirements: SRS_BSW_00323, SRS_BSW_00357, SRS_BSW_00449, SRS_SecOC_00017

⌈

Service Name	SecOC_VerifyStatusOverride	
Syntax	`Std_ReturnType SecOC_VerifyStatusOverride ( uint16 ValueID, SecOC_OverrideStatusType overrideStatus, uint8 numberOfMessagesToOverride )`	
Service ID [hex]	0x0b	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant for the same FreshnessValueID. Reentrant for different FreshnessValueIDs	
Parameters (in)	ValueID	If SecOCOverrideStatusWithDataId is configured to FALSE, ValueID is the ID of the Freshness Value used to control the verification behaviour of all assigned Secured I-PDUs according to the overrideStatus. If SecOCOverrideStatusWithDataId is configured to TRUE, ValueID is the DataID of a Secured I-PDU that shall be controlled by the overrideStatus.
	overrideStatus	Defines whether verification is executed and whether the I-PDU is passed on, and for how long the override is active.
	numberOfMessagesTo Override	Number of sequential verification to override when using a specific counter for authentication verification. This is only considered when OverrideStatus is equal to SECOC_ OVERRIDE_DROP_UNTIL_LIMIT, SECOC_OVERRIDE_SKIP_ UNTIL_LIMIT or SECOC_OVERRIDE_PASS_UNTIL_LIMIT.
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: request successful `E_NOT_OK`: request failed
Description	This service provides the ability to force specific behaviour of SecOc: accept or drop an I-PDU with or without performing the verification of authenticator or independent of the authenticator verification result, and to force a specific result for SecOC_VerificationResultType allowing additional fault handling in the application. Option SECOC_OVERRIDE_PASS_UNTIL_ NOTICE, SECOC_OVERRIDE_SKIP_UNTIL_LIMIT, SECOC_OVERRIDE_PASS_UNTIL_ LIMIT or SECOC_OVERRIDE_SKIP_UNTIL_NOTICE are available only if SecOCEnable ForcedPassOverride is set to TRUE.	
Available via	SecOC.h	

⌋

8.3.10 SecOC_SendDefaultAuthenticationInformation

[SWS_SecOC_91013] Definition of API function SecOC_SendDefaultAuthenticationInformation

Upstream requirements: SRS_SecOC_00021

⌈

Service Name	SecOC_SendDefaultAuthenticationInformation	
Syntax	`Std_ReturnType SecOC_SendDefaultAuthenticationInformation ( uint16 FreshnessValueID, boolean sendDefaultAuthenticationInformation )`	
Service ID [hex]	0x04	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant for the same FreshnessValueID. Reentrant for different FreshnessValueIDs	
Parameters (in)	FreshnessValueID	ID of the Freshness Value for which sending SecOCDefault AuthenticationInformationPattern should be enabled.
	sendDefault AuthenticationInformation	FALSE - sending SecOCDefaultAuthenticationInformationPattern shall be disabled for given FreshnessValueID TRUE - sending SecOCDefaultAuthenticationInformationPattern shall be enabled for given FreshnessValueID
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: request successful `E_NOT_OK`: request failed
Description	The service provides the ability to enable the sending of un-authenticated PDU to lower layer. (example: in case authentication build counter has reached the configuration value Sec OCAuthenticationBuildAttempts or the query of the freshness function returns E_NOT_OK or the calculation of the authenticator has returned a non-recoverable error such as returning E_ NOT_OK or KEY_FAILURE). This service is optional (the service is available only if Sec OCDefaultAuthenticationInformationPattern is configured). If the service is not available or the service is available but the service was called with sendDefaultAuthenticationInformation as FALSE for a given FreshnessValueID , SecOC module shall remove the Authentic I-PDU from its internal buffer and cancel the transmission request in case the building of authentication Information failed. If the service is available and the service was called with sendDefault AuthenticationInformation as TRUE for a given FreshnessValueID , SecOc will use Sec OCDefaultAuthenticationInformationPattern as authentication Information and will not cancel the transmission request.	
Available via	SecOC.h	

⌋

8.4 Callback notifications

8.4.1 SecOC_RxIndication

[SWS_SecOC_00124] Definition of callback function SecOC_RxIndication

Upstream requirements: SRS_BSW_00323, SRS_BSW_00359, SRS_SecOC_00012

⌈

Service Name	SecOC_RxIndication	

Syntax	`void SecOC_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.	
Available via	SecOC.h	

⌋

8.4.2 SecOC_TpRxIndication

[SWS_SecOC_00125] Definition of callback function SecOC_TpRxIndication

Upstream requirements: SRS_BSW_00323, SRS_BSW_00359, SRS_BSW_00449, SRS_SecOC_00012

⌈

Service Name	SecOC_TpRxIndication	

Syntax	`void SecOC_TpRxIndication (`
	`  PduIdType id,`
	`  Std_ReturnType result`
	`)`	

Service ID [hex]	0x45	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	id	Identification of the received I-PDU.

▽

△

	result	E_OK: The PDU was received. E_NOT_OK: Reception of the PDU failed.
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	Called after an I-PDU has been received via the TP API, the result indicates whether the transmission was successful or not.	
Available via	SecOC.h	

⌋

8.4.3 SecOC_TxConfirmation

[SWS_SecOC_00126] Definition of callback function SecOC_TxConfirmation

Upstream requirements: SRS_BSW_00323, SRS_BSW_00359, SRS_SecOC_00012

⌈

Service Name	SecOC_TxConfirmation	
Syntax	`void SecOC_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. 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.	
Available via	SecOC.h	

⌋

8.4.4 SecOC_TpTxConfirmation

[SWS_SecOC_00152] Definition of callback function SecOC_TpTxConfirmation

Upstream requirements: SRS_BSW_00323, SRS_BSW_00359, SRS_BSW_00449, SRS_SecOC_00012

⌈

Service Name	SecOC_TpTxConfirmation	

Syntax	`void SecOC_TpTxConfirmation (`
	`  PduIdType id,`
	`  Std_ReturnType result`
	`)`	

Service ID [hex]	0x48	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	id	Identification of the transmitted I-PDU.
	result	E_OK: The PDU was transmitted. E_NOT_OK: Transmission of the PDU failed.
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	This function is called after the I-PDU has been transmitted on its network, the result indicates whether the transmission was successful or not.	
Available via	SecOC.h	

⌋

8.4.5 SecOC_TriggerTransmit

[SWS_SecOC_00127] Definition of callback function SecOC_TriggerTransmit

Upstream requirements: SRS_BSW_00323, SRS_BSW_00357, SRS_BSW_00449, SRS_SecOC_00012

⌈

Service Name	SecOC_TriggerTransmit	

Syntax	`Std_ReturnType SecOC_TriggerTransmit (`
	`  PduIdType TxPduId,`
	`  PduInfoType* PduInfoPtr`
	`)`	

Service ID [hex]	0x41	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different PduIds. Non reentrant for the same PduId.	
Parameters (in)	TxPduId	ID of the SDU that is requested to be transmitted.

▽

△

Parameters (inout)	PduInfoPtr	Contains a pointer to a buffer (SduDataPtr) to where the SDU data shall be copied, and the available buffer size in SduLengh. On return, the service will indicate the length of the copied SDU data in SduLength.
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: SDU has been copied and SduLength indicates the number of copied bytes. `E_NOT_OK`: No SDU data has been copied. PduInfoPtr must not be used since it may contain a NULL pointer or point to invalid data.
Description	Within this API, the upper layer module (called module) shall check whether the available data fits into the buffer size reported by PduInfoPtr->SduLength. If it fits, it shall copy its data into the buffer provided by PduInfoPtr->SduDataPtr and update the length of the actual copied data in PduInfoPtr->SduLength. If not, it returns E_NOT_OK without changing PduInfoPtr.	
Available via	SecOC.h	

⌋

8.4.6 SecOC_CopyRxData

[SWS_SecOC_00128] Definition of callback function SecOC_CopyRxData

Upstream requirements: SRS_BSW_00323, SRS_BSW_00357, SRS_SecOC_00012

⌈

Service Name	SecOC_CopyRxData	

Syntax	`BufReq_ReturnType SecOC_CopyRxData (`
	`  PduIdType id,`
	`  const PduInfoType* info,`
	`  PduLengthType* bufferSizePtr`
	`)`	

Service ID [hex]	0x44	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	id	Identification of the received I-PDU.
	info	Provides the source buffer (SduDataPtr) and the number of bytes to be copied (SduLength). An SduLength of 0 can be used to query the current amount of available buffer in the upper layer module. In this case, the SduDataPtr may be a NULL_PTR.
Parameters (inout)	None	
Parameters (out)	bufferSizePtr	Available receive buffer after data has been copied.
Return value	BufReq_ReturnType	`BUFREQ_OK`: Data copied successfully `BUFREQ_E_NOT_OK`: Data was not copied because an error occurred.
Description	This function is called to provide the received data of an I-PDU segment (N-PDU) to the upper layer. Each call to this function provides the next part of the I-PDU data. The size of the remaining buffer is written to the position indicated by bufferSizePtr.	
Available via	SecOC.h	

⌋

8.4.7 SecOC_CopyTxData

[SWS_SecOC_00129] Definition of callback function SecOC_CopyTxData

Upstream requirements: SRS_BSW_00323, SRS_BSW_00357, SRS_SecOC_00012

⌈

Service Name	SecOC_CopyTxData	

Syntax	`BufReq_ReturnType SecOC_CopyTxData (`
	`  PduIdType id,`
	`  const PduInfoType* info,`
	`  const RetryInfoType* retry,`
	`  PduLengthType* availableDataPtr`
	`)`	

Service ID [hex]	0x43	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	id	Identification of the transmitted I-PDU.
	info	Provides the destination buffer (SduDataPtr) and the number of bytes to be copied (SduLength). If not enough transmit data is available, no data is copied by the upper layer module and BUFREQ_E_BUSY is returned. The lower layer module may retry the call. An SduLength of 0 can be used to indicate state changes in the retry parameter or to query the current amount of available data in the upper layer module. In this case, the Sdu DataPtr may be a NULL_PTR.
	retry	This parameter is used to acknowledge transmitted data or to retransmit data after transmission problems. If the retry parameter is a NULL_PTR, it indicates that the transmit data can be removed from the buffer immediately after it has been copied. Otherwise, the retry parameter must point to a valid RetryInfoType element. If TpDataState indicates TP_CONFPENDING, the previously copied data must remain in the TP buffer to be available for error recovery. TP_DATACONF indicates that all data that has been copied before this call is confirmed and can be removed from the TP buffer. Data copied by this API call is excluded and will be confirmed later. TP_DATARETRY indicates that this API call shall copy previously copied data in order to recover from an error. In this case TxTpDataCnt specifies the offset in bytes from the current data copy position.
Parameters (inout)	None	
Parameters (out)	availableDataPtr	Indicates the remaining number of bytes that are available in the upper layer module's Tx buffer. availableDataPtr can be used by TP modules that support dynamic payload lengths (e.g. FrIsoTp) to determine the size of the following CFs.
Return value	BufReq_ReturnType	`BUFREQ_OK`: Data has been copied to the transmit buffer completely as requested. `BUFREQ_E_BUSY`: Request could not be fulfilled, because the required amount of Tx data is not available. The lower layer module may retry this call later on. No data has been copied. `BUFREQ_E_NOT_OK`: Data has not been copied. Request failed.

▽

△

Description	This function is called to acquire the transmit data of an I-PDU segment (N-PDU). Each call to this function provides the next part of the I-PDU data unless retry->TpDataState is TP_DATARETRY. In this case the function restarts to copy the data beginning at the offset from the current position indicated by retry->TxTpDataCnt. The size of the remaining data is written to the position indicated by availableDataPtr.
Available via	SecOC.h

⌋

8.4.8 SecOC_StartOfReception

[SWS_SecOC_00130] Definition of callback function SecOC_StartOfReception

Upstream requirements: SRS_BSW_00323, SRS_BSW_00357, SRS_SecOC_00012

⌈

Service Name	SecOC_StartOfReception	
Syntax	`BufReq_ReturnType SecOC_StartOfReception ( PduIdType id, const PduInfoType* info, PduLengthType TpSduLength, PduLengthType* bufferSizePtr )`	
Service ID [hex]	0x46	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	id	Identification of the I-PDU.
	info	Pointer to a PduInfoType structure containing the payload data (without protocol information) and payload length of the first frame or single frame of a transport protocol I-PDU reception, and the MetaData related to this PDU. If neither first/single frame data nor MetaData are available, this parameter is set to NULL_PTR.
	TpSduLength	Total length of the N-SDU to be received.
Parameters (inout)	None	
Parameters (out)	bufferSizePtr	Available receive buffer in the receiving module. This parameter will be used to compute the Block Size (BS) in the transport protocol module.
Return value	BufReq_ReturnType	`BUFREQ_OK`: Connection has been accepted. bufferSizePtr indicates the available receive buffer; reception is continued. If no buffer of the requested size is available, a receive buffer size of 0 shall be indicated by bufferSizePtr. `BUFREQ_E_NOT_OK`: Connection has been rejected; reception is aborted. bufferSizePtr remains unchanged. `BUFREQ_E_OVFL`: No buffer of the required length can be provided; reception is aborted. bufferSizePtr remains unchanged.
Description	This function is called at the start of receiving an N-SDU. The N-SDU might be fragmented into multiple N-PDUs (FF with one or more following CFs) or might consist of a single N-PDU (SF).	
Available via	SecOC.h	

⌋

[SWS_SecOC_00181]

Upstream requirements: SRS_BSW_00385, SRS_SecOC_00012

⌈In case `SecOC_StartOfReception` is called with `TpSduLength` equal to 0, the SecOC module shall return `BUFREQ_E_NOT_OK` and no further action shall be taken.⌋

8.4.9 CSM callback interfaces

[SWS_SecOC_00012]

Upstream requirements: SRS_BSW_00457, SRS_SecOC_00003

⌈If the SecOC module uses the Csm module asynchronously to calculate or verify the authenticator, SecOC shall provide adequate callback functions for every CsmJob to get notification about the result of the asynchronous job.⌋

Note: CSM jobs can run synchronously or asynchronously, which depends on its configuration. For asynchronous jobs a callback is needed to get notified when the operation is finished. This callback is not defined in this document. They are vendor specific and shall be configured accordingly in the CSM as documented in [SWS_Csm_00971].

8.5 Callout Definitions

Callouts are pieces of code that have to be added to the SecOC during ECU integration. The content of most callouts is hand-written code.

8.5.1 SecOC_GetRxFreshness

[SWS_SecOC_91007] Definition of API function SecOC_GetRxFreshness

Upstream requirements: SRS_SecOC_00003

⌈

Service Name	SecOC_GetRxFreshness

Syntax	`Std_ReturnType SecOC_GetRxFreshness (`
	`  uint16 SecOCFreshnessValueID,`
	`  const uint8* SecOCTruncatedFreshnessValue,`
	`  uint32 SecOCTruncatedFreshnessValueLength,`
	`  uint16 SecOCAuthVerifyAttempts,`
	`  uint8* SecOCFreshnessValue,`
	`  uint32* SecOCFreshnessValueLength`
	`)`

Service ID [hex]	0x4f

▽

△

Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	SecOCFreshnessValueID	Holds the identifier of the freshness value.
	SecOCTruncated FreshnessValue	Holds the truncated freshness value that was contained in the Secured I-PDU.
	SecOCTruncated FreshnessValueLength	Holds the length in bits of the truncated freshness value.
	SecOCAuthVerify Attempts	Holds the number of authentication verify attempts of this PDU since the last reception. The value is 0 for the first attempt and incremented on every unsuccessful verification attempt.
Parameters (inout)	SecOCFreshnessValue Length	Holds the length in bits of the freshness value.
Parameters (out)	SecOCFreshnessValue	Holds the freshness value to be used for the calculation of the authenticator.
Return value	Std_ReturnType	`E_OK`: request successful `E_NOT_OK`: request failed, a freshness value cannot be provided due to general issues for freshness or this FreshnessValueId. `E_BUSY`: The freshness information can temporarily not be provided.
Description	This interface is used by the SecOC to obtain the current freshness value.	
Available via	SecOC.h	

⌋

8.5.2 SecOC_GetRxFreshnessAuthData

[SWS_SecOC_91006] Definition of API function SecOC_GetRxFreshnessAuthData

Upstream requirements: SRS_SecOC_00003

⌈

Service Name	SecOC_GetRxFreshnessAuthData	

Syntax	`Std_ReturnType SecOC_GetRxFreshnessAuthData (`
	`  uint16 SecOCFreshnessValueID,`
	`  const uint8* SecOCTruncatedFreshnessValue,`
	`  uint32 SecOCTruncatedFreshnessValueLength,`
	`  const uint8* SecOCAuthDataFreshnessValue,`
	`  uint16 SecOCAuthDataFreshnessValueLength,`
	`  uint16 SecOCAuthVerifyAttempts,`
	`  uint8* SecOCFreshnessValue,`
	`  uint32* SecOCFreshnessValueLength`
	`)`	

Service ID [hex]	0x4e	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	SecOCFreshnessValueID	Holds the identifier of the freshness value.
	SecOCTruncated FreshnessValue	Holds the truncated freshness value that was contained in the Secured I-PDU.

▽

△

	SecOCTruncated FreshnessValueLength	Holds the length in bits of the truncated freshness value.
	SecOCAuthData FreshnessValue	The parameter holds a part of the received, not yet authenticated PDU. The parameter is optional (see description)
	SecOCAuthData FreshnessValueLength	This is the length value in bits that holds the freshness from the authentic PDU. The parameter is optional (see description).
	SecOCAuthVerify Attempts	Holds the number of authentication verify attempts of this PDU since the last reception. The value is 0 for the first attempt and incremented on every unsuccessful verification attempt.
Parameters (inout)	SecOCFreshnessValue Length	Holds the length in bits of the freshness value.
Parameters (out)	SecOCFreshnessValue	Holds the freshness value to be used for the calculation of the authenticator.
Return value	Std_ReturnType	`E_OK`: request successful `E_NOT_OK`: request failed, a freshness value cannot be provided due to general issues for freshness or this FreshnessValueId. `E_BUSY`: The freshness information can temporarily not be provided.
Description	This interface is used by the SecOC to obtain the current freshness value.	
Available via	SecOC.h	

⌋

8.5.3 SecOC_GetTxFreshness

[SWS_SecOC_91004] Definition of API function SecOC_GetTxFreshness

Upstream requirements: SRS_SecOC_00003, SRS_SecOC_00006

⌈

Service Name	SecOC_GetTxFreshness	
Syntax	`Std_ReturnType SecOC_GetTxFreshness ( uint16 SecOCFreshnessValueID, uint8* SecOCFreshnessValue, uint32* SecOCFreshnessValueLength )`	
Service ID [hex]	0x52	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	SecOCFreshnessValueID	Holds the identifier of the freshness value.
Parameters (inout)	SecOCFreshnessValue Length	Holds the length of the provided freshness in bits.
Parameters (out)	SecOCFreshnessValue	Holds the current freshness value
Return value	Std_ReturnType	`E_OK`: request successful `E_NOT_OK`: request failed, a freshness value cannot be provided due to general issues for freshness or this FreshnessValueId. `E_BUSY`: The freshness information can temporarily not be provided.

▽

△

Description	This API returns the freshness value from the Most Significant Bits in the first byte in the array (SecOCFreshnessValue), in big endian format.
Available via	SecOC.h

⌋

8.5.4 SecOC_GetTxFreshnessTruncData

[SWS_SecOC_91003] Definition of API function SecOC_GetTxFreshnessTruncData

Upstream requirements: SRS_SecOC_00003, SRS_SecOC_00006

⌈

Service Name	SecOC_GetTxFreshnessTruncData	
Syntax	`Std_ReturnType SecOC_GetTxFreshnessTruncData ( uint16 SecOCFreshnessValueID, uint8* SecOCFreshnessValue, uint32* SecOCFreshnessValueLength, uint8* SecOCTruncatedFreshnessValue, uint32* SecOCTruncatedFreshnessValueLength )`	
Service ID [hex]	0x51	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	SecOCFreshnessValueID	Holds the identifier of the freshness value.
Parameters (inout)	SecOCFreshnessValue Length	Holds the length of the provided freshness in bits.
	SecOCTruncated FreshnessValueLength	Provides the truncated freshness length configured for this freshness. The function may adapt the value if needed or can leave it unchanged if the configured length and provided length is the same.
Parameters (out)	SecOCFreshnessValue	Holds the current freshness value.
	SecOCTruncated FreshnessValue	Holds the truncated freshness to be included into the Secured I-PDU. The parameter is optional.
Return value	Std_ReturnType	`E_OK`: request successful `E_NOT_OK`: request failed, a freshness value cannot be provided due to general issues for freshness or this FreshnessValueId. `E_BUSY`: The freshness information can temporarily not be provided.
Description	This interface is used by the SecOC to obtain the current freshness value. The interface function provides also the truncated freshness transmitted in the secured I-PDU.	
Available via	SecOC.h	

⌋

8.5.5 SecOC_SPduTxConfirmation

[SWS_SecOC_91005] Definition of API function SecOC_SPduTxConfirmation

Upstream requirements: SRS_SecOC_00002, SRS_SecOC_00003

⌈

Service Name	SecOC_SPduTxConfirmation	
Syntax	`void SecOC_SPduTxConfirmation ( uint16 SecOCFreshnessValueID )`	
Service ID [hex]	0x4d	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	SecOCFreshnessValueID	Holds the identifier of the freshness value.
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	This interface is used by the SecOC to indicate that the Secured I-PDU has been initiated for transmission.	
Available via	SecOC.h	

⌋

8.6 Scheduled functions

8.6.1 SecOC_MainFunctionRx

[SWS_SecOC_00171] Definition of scheduled function SecOC_MainFunctionRx

Upstream requirements: SRS_BSW_00373, SRS_BSW_00425

⌈

Service Name	SecOC_MainFunctionRx
Syntax	`void SecOC_MainFunctionRx ( void )`
Service ID [hex]	0x06
Description	This function performs the processing of the SecOC module's authentication and verification processing for the Rx path. Per configured SecOCMainFunctionRx instance one SecOC_Main FunctionRx_<shortName> shall be implemented. Hereby <shortName> is the short name of the SecOCMainFunctionRx configuration container in the ECU configuration.
Available via	SchM_SecOC.h

⌋

[SWS_SecOC_00172]

Upstream requirements: SRS_SecOC_00005

⌈If the SecOC module was not previously initialized with a call to `SecOC_Init`, then a call to `SecOC_MainFunctionRx` shall simply return.⌋

[SWS_SecOC_00173]

Upstream requirements: SRS_SecOC_00025

⌈The cycle time of the `SecOC_MainFunctionRx` is configured by the parameter `SecOCMainFunctionPeriodRx`.⌋

[SWS_SecOC_00174]

Upstream requirements: SRS_SecOC_00025

⌈If `SecOC_MainFunctionRx` is scheduled, the SecOC shall firstly check if there are new Secured I-PDUs to be verified. If yes the SecOC module shall process the verification of each of the IPDUs identified as new subsequently in the very same main function call.⌋

[SWS_SecOC_00175]

Upstream requirements: SRS_SecOC_00025

⌈For each newly successfully verified Secured I-PDU, the SecOC module shall immediately pass the Authentic I-PDU to the upper layer communication module by calling `PduR_SecOC[If|Tp]RxIndication` for the Authentic I-PDU.⌋

8.6.2 SecOC_MainFunctionTx

[SWS_SecOC_00176] Definition of scheduled function SecOC_MainFunctionTx

Upstream requirements: SRS_BSW_00373, SRS_BSW_00425

⌈

Service Name	SecOC_MainFunctionTx

Syntax	`void SecOC_MainFunctionTx (`
	`  void`
	`)`

Service ID [hex]	0x03
Description	This function performs the processing of the SecOC module's authentication and verification processing for the Tx path. Per configured SecOCMainFunctionTx instance one SecOC_Main FunctionTx_<shortName> shall be implemented. Hereby <shortName> is the short name of the SecOCMainFunctionTx configuration container in the ECU configuration.
Available via	SchM_SecOC.h

⌋

[SWS_SecOC_00177]

Upstream requirements: SRS_SecOC_00005

⌈If the SecOC module was not previously initialized with a call to `SecOC_Init`, then a call to `SecOC_MainFunctionTx` shall simply return.⌋

[SWS_SecOC_00178]

Upstream requirements: SRS_SecOC_00025

⌈The cycle time of the `SecOC_MainFunctionTx` is configured by the parameter `SecOCMainFunctionPeriodTx`.⌋

[SWS_SecOC_00179]

Upstream requirements: SRS_SecOC_00025

⌈If `SecOC_MainFunctionTx` is scheduled, the SecOC shall firstly check if there are new Authentic I-PDUs to be authenticated. If yes the SecOC module shall process the authentication of each of the IPDUs identified as new subsequently in the very same main function call.⌋

[SWS_SecOC_00180]

Upstream requirements: SRS_SecOC_00025

⌈For each newly authenticated Authentic I-PDU, the SecOC module shall immediately trigger the transmission of the Secured I-PDU at the lower layer module by calling the PduR.⌋

8.7 Expected interfaces

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

8.7.1 Mandatory interfaces

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

[SWS_SecOC_00137] Definition of mandatory interfaces required by module SecOC

Upstream requirements: SRS_BSW_00384

⌈

API Function	Header File	Description
Det_ReportRuntimeError	Det.h	Service to report runtime errors. If a callout has been configured then this callout shall be called.
PduR_SecOCCancelTransmit	PduR_SecOC.h	Requests cancellation of an ongoing transmission of a PDU in a lower layer communication module.
PduR_SecOCIfRxIndication	PduR_SecOC.h	Indication of a received PDU from a lower layer communication interface module.
PduR_SecOCIfTxConfirmation	PduR_SecOC.h	The lower layer communication interface module confirms the transmission of a PDU, or the failure to transmit a PDU.
PduR_SecOCTransmit	PduR_SecOC.h	Requests transmission of a PDU.

⌋

8.7.2 Optional interfaces

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

[SWS_SecOC_00138] Definition of optional interfaces requested by module SecOC

Upstream requirements: SRS_BSW_00384

⌈

API Function	Header File	Description
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_SignatureGenerate	Csm.h	Uses the given data to perform the signature calculation and stores the signature in the memory location pointed by the result pointer.
Csm_SignatureVerify	Csm.h	Verifies the given signature by checking if it was generated with the given data.
Det_ReportError	Det.h	Service to report development errors.
IdsM_SetSecurityEvent (obsolete)	IdsM.h	This API is the application interface to report security events to the IdsM. **Tags:** atp.Status=obsolete

▽

△

API Function	*Header File*	*Description*
IdsM_SetSecurityEventWithContext Data (obsolete)	IdsM.h	This API is the application interface to report security events with context data to the IdsM. **Tags:** atp.Status=obsolete
PduR_SecOCCancelReceive	PduR_SecOC.h	Requests cancellation of an ongoing reception of a PDU in a lower layer transport protocol module.
PduR_SecOCTpCopyRxData	PduR_SecOC.h	This function is called to provide the received data of an I-PDU segment (N-PDU) to the upper layer. Each call to this function provides the next part of the I-PDU data. The size of the remaining buffer is written to the position indicated by bufferSizePtr.
PduR_SecOCTpCopyTxData	PduR_SecOC.h	This function is called to acquire the transmit data of an I-PDU segment (N-PDU). Each call to this function provides the next part of the I-PDU data unless retry->TpDataState is TP_DATARETRY. In this case the function restarts to copy the data beginning at the offset from the current position indicated by retry->TxTpDataCnt. The size of the remaining data is written to the position indicated by availableDataPtr.
PduR_SecOCTpRxIndication	PduR_SecOC.h	Called after an I-PDU has been received via the TP API, the result indicates whether the transmission was successful or not.
PduR_SecOCTpStartOfReception	PduR_SecOC.h	This function is called at the start of receiving an N-SDU. The N-SDU might be fragmented into multiple N-PDUs (FF with one or more following CFs) or might consist of a single N-PDU (SF). The service shall provide the currently available maximum buffer size when invoked with TpSdu Length equal to 0.
PduR_SecOCTpTxConfirmation	PduR_SecOC.h	This function is called after the I-PDU has been transmitted on its network, the result indicates whether the transmission was successful or not.

⌋

8.7.3 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.

8.7.3.1 SecOC_VerificationStatusCallout

If configured by `SecOCVerificationStatusCallout` (see [ECUC_SecOC_00004]), the SecOC module shall invoke a callout function to notify other modules on the verification status of the most recently received Secured I-PDU.

[SWS_SecOC_00119] Definition of configurable interface SecOC_VerificationStatusCallout

Upstream requirements: SRS_BSW_00359, SRS_SecOC_00017

⌈

Service Name	SecOC_VerificationStatusCallout	
Syntax	`void SecOC_VerificationStatusCallout ( SecOC_VerificationStatusType verificationStatus )`	
Service ID [hex]	0x50	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant for the same FreshnessValueID. Reentrant for different FreshnessValueIDs	
Parameters (in)	verificationStatus	Data structure to bundle the status of a verification attempt for a specific Freshness Value and Data ID
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	Service is used to propagate the status of each verification attempt from the SecOC module to other modules. This service can be configured such that: • Only: "False" Verification Status is propagated to modules • Both: "True" and "False" Verification Status are propagated to modules • None: No Verification Status is propagated	
Available via	SecOC_Externals.h	

⌋

Note: The argument freshnessValueID allows for unambiguously identifying the Secured I-PDU that was subject of the verification attempt. Since each Secured I-PDU has at least one but possibly two related Freshness Value IDs (i.e. a Secured I-PDU may have a Secondary Freshness Value ID), `SecOC_VerificationStatusCallout` is able to indicate for which of the freshness values the verification attempt has been carried out.

Note: Any module that is configured to be notified by the means of `SecOC_VerificationStatusCallout` has to implement a target function that is conforming to the above signature. The name of the target function listed above are not fixed. The name could be configured by means of the parameter `SecOCVerificationStatusCallout`.

8.7.3.2 SecOC_VerifyStatus

[SWS_SecOC_91014] Definition of API function SecOC_VerifyStatus

Upstream requirements: SRS_SecOC_00003, SRS_SecOC_00029

⌈

Service Name	SecOC_VerifyStatus	
Syntax	`void SecOC_VerifyStatus ( SecOC_VerificationStatusType verificationStatus )`	
Service ID [hex]	0x55	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant for the same FreshnessValueID. Reentrant for different FreshnessValueIDs	
Parameters (in)	verificationStatus	The verificationStatus is a structure that provides details about the verification status and on which DataId and FreshnessValueId the verification was performed.
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	This service provides the ability to inform the application about the result of the verification attempt of a received PDU by the SecOC module.	
Available via	SecOC_Externals.h	

⌋

8.8 Service Interfaces

This chapter defines the AUTOSAR Interfaces of the SecOC Service (<MA>).

The definitions in this section are interpreted to be in ARPackage AUTOSAR/Services/<MA>.

8.8.1 Overview

This chapter is an addition to the specification of the SecOC module. Whereas the other parts of the specification define the behavior and the C-interfaces of the corresponding basic software module, this chapter formally specifies the corresponding AUTOSAR service in terms of the SWC template. The interfaces described here will be visible on the VFB and are used to generate the Rte between application software and the SecOC module.

8.8.2 Sender-Receiver-Interfaces

8.8.2.1 Verification Status Service

[SWS_SecOC_00141] Definition of SenderReceiverInterface VerificationStatus

Upstream requirements: SRS_SecOC_00022

⌈

Name	VerificationStatus	
Comment	This service realizes a notification service that is used to propagate the status of each authentication attempt from the SecOC module to the application layer. This service can be configured such that: • Only "False" Verification Status is propagated to the application layer • Both "True" and "False" Verification Status are propagated to the application layer • No Verification Status is propagated to the application layer	
IsService	true	
Variation	–	
Data Elements	verificationStatus	
	Type	SecOC_VerificationStatusType
	Variation	–

⌋

Note: The `VerificationStatus` is used to propagate the status of each verification attempt from the SecOC module to an arbitrary number of application software components. It can be used to continuously monitor the number of failed verification attempts and would allow setting up a security management system/intrusion detection system that is able to detect an attack flood and react with adequate dynamic countermeasures.

[SWS_SecOC_00148]

Upstream requirements: SRS_SecOC_00022

⌈SecOC shall define a provide port for the `VerificationStatus` interface and call the generated Rte function as configured by the parameter `SecOCVerificationStatusPropagationMode`. The sender/receiver interface shall be defined as standard interface.⌋

8.8.3 Client-Server-Interfaces

8.8.3.1 Verification Status Configuration Service

[SWS_SecOC_00142] Definition of ClientServerInterface VerifyStatusConfiguration

Upstream requirements: SRS_SecOC_00017

⌈

Name	VerifyStatusConfiguration		
Comment	Verify Status Configuration Service of SecOC		
IsService	true		
Variation	–		
Possible Errors	0	E_OK	Operation successful
	1	E_NOT_OK	Operation failed

Operation	VerifyStatusOverride	
Comment	This service provides the ability to force specific behaviour of SecOc: accept or drop an I-PDU with or without performing the verification of authenticator or independent of the authenticator verification result, and to force a specific result for SecOC_VerificationResultType allowing additional fault handling in the application. Option SECOC_OVERRIDE_PASS_UNTIL_NOTICE, SECOC_OVERRIDE_SKIP_UNTIL_LIMIT, SECOC_OVERRIDE_PASS_UNTIL_LIMIT or SECOC_OVERRIDE_SKIP_UNTIL_NOTICE are available only if SecOCEnableForcedPassOverride is set to TRUE.	
Mapped to API	SecOC_VerifyStatusOverride	
Variation	–	
Parameters	ValueId	
	Type	uint16
	Direction	IN
	Comment	Identifier of the Value ID where override shall be applied to. If configuration option SecOCOverrideStatusWithDataId is set to TRUE, this value shall provide the DataID of the secured I-PDU. If SecOCOverrideStatusWithDataId is set to FALSE, this parameter shall provide the freshness value ID.
	Variation	–
	overrideStatus	
	Type	SecOC_OverrideStatusType
	Direction	IN
	Comment	Defines whether verification is executed and whether the I-PDU is passed on, and for how long the override is active.
	Variation	–
	numberOfMessagesToOverride	
	Type	uint8
	Direction	IN
	Comment	Number of sequential VerifyStatus to override when using a specific counter for authentication verification. This is only considered when OverrideStatus is equal to SECOC_OVERRIDE_DROP_UNTIL_LIMIT, SECOC_OVERRIDE_SKIP_UNTIL_LIMIT or SECOC_OVERRIDE_PASS_UNTIL_LIMIT.
	Variation	–

▽

△

Possible Errors	E_OK E_NOT_OK

⌋

8.8.3.2 FreshnessManagement

[SWS_SecOC_91002] Definition of ClientServerInterface FreshnessManagement

Upstream requirements: SRS_SecOC_00003, SRS_SecOC_00021, SRS_SecOC_00022

⌈

Name	FreshnessManagement		
Comment	Freshness Management for SecOC		
IsService	true		
Variation	–		
Possible Errors	0	E_OK	Operation successful
	1	E_NOT_OK	Operation failed
	2	E_BUSY	Operation temporary failed, a freshness cannot be provided at the moment.

Operation	GetRxFreshness	
Comment	This interface is used by the SecOC to obtain the current freshness value. This operation provides also a part of the Authentic-PDU data if configured.	
Mapped to API	SecOC_GetRxFreshness	
Variation	({ecuc(SecOC/SecOCRxPduProcessing/SecOCUseAuthDataFreshness)} == FALSE)	
Parameters	freshnessValueId	
	Type	uint16
	Direction	IN
	Comment	Identifier of the freshness
	Variation	–
	truncatedFreshnessValue	
	Type	SecOC_FreshnessArrayType
	Direction	IN
	Comment	The truncated freshness value from the received Secured-IPDU
	Variation	–
	truncatedFreshnessValueLength	
	Type	uint32
	Direction	IN
	Comment	Length in bits of the truncated freshness value
	Variation	–
	authVerifyAttempts	
	Type	uint16
	Direction	IN

▽

△

	Comment	The number of authentication verify attempts for the current PDU
	Variation	–
	freshnessValue	
	Type	SecOC_FreshnessArrayType
	Direction	OUT
	Comment	The freshness value for this PDU
	Variation	–
	freshnessValueLength	
	Type	uint32
	Direction	INOUT
	Comment	The freshness value length in bits.
	Variation	–
Possible Errors	E_OK E_NOT_OK E_BUSY	

Operation	GetRxFreshnessAuthData	
Comment	This interface is used by the SecOC to obtain the current freshness value. This operation provides also a part of the Authentic-PDU data if configured.	
Mapped to API	SecOC_GetRxFreshnessAuthData	
Variation	({ecuc(SecOC/SecOCRxPduProcessing/SecOCUseAuthDataFreshness)} == TRUE)	
Parameters	freshnessValueId	
	Type	uint16
	Direction	IN
	Comment	Identifier of the freshness
	Variation	–
	truncatedFreshnessValue	
	Type	SecOC_FreshnessArrayType
	Direction	IN
	Comment	The truncated freshness value from the received Secured-IPDU
	Variation	–
	truncatedFreshnessValueLength	
	Type	uint32
	Direction	IN
	Comment	Length in bits of the truncated freshness value
	Variation	–
	authenticDataFreshnessValue	
	Type	SecOC_FreshnessArrayType
	Direction	IN
	Comment	The selected part of the authentic data.
	Variation	–
	authenticDataFreshnessValueLength	
	Type	uint16
	Direction	IN
	Comment	The length in bits of the authentic data part.
	Variation	–
	authVerifyAttempts	

▽

△

	Type	uint16
	Direction	IN
	Comment	The number of authentication verify attempts for this PDU
	Variation	–
	freshnessValue	
	Type	SecOC_FreshnessArrayType
	Direction	OUT
	Comment	The freshness value for this PDU
	Variation	–
	freshnessValueLength	
	Type	uint32
	Direction	INOUT
	Comment	The freshness value length in bits.
	Variation	–
Possible Errors	E_OK E_NOT_OK E_BUSY	

Operation	GetTxFreshness	
Comment	Returns the freshness value from the Most Significant Bits in the first byte in the array (Sec OCFreshnessValue), in big endian format.	
Mapped to API	SecOC_GetTxFreshness	
Variation	({ecuc(SecOC/SecOCTxPduProcessing/SecOCProvideTxTruncatedFreshnessValue)} == FALSE)	
Parameters	freshnessValueId	
	Type	uint16
	Direction	IN
	Comment	Identifier of the freshness
	Variation	–
	freshnessValue	
	Type	SecOC_FreshnessArrayType
	Direction	OUT
	Comment	Freshness value
	Variation	–
	freshnessValueLength	
	Type	uint32
	Direction	INOUT
	Comment	Length in bits of the freshness value
	Variation	–
Possible Errors	E_OK E_NOT_OK E_BUSY	

Operation	GetTxFreshnessTruncData
Comment	This operation is used by the SecOC to obtain the freshness that corresponds to the freshness ValueId. The operation provides the freshness and also the truncated freshness that shall be placed into the Secured-IPDU.
Mapped to API	SecOC_GetTxFreshnessTruncData

▽

△

Variation	({ecuc(SecOC/SecOCTxPduProcessing/SecOCProvideTxTruncatedFreshnessValue)} == TRUE)	
Parameters	freshnessValueId	
	Type	uint16
	Direction	IN
	Comment	Identifier of the freshness
	Variation	–
	freshnessValue	
	Type	SecOC_FreshnessArrayType
	Direction	OUT
	Comment	Freshness value
	Variation	–
	freshnessValueLength	
	Type	uint32
	Direction	INOUT
	Comment	Length in bits of the freshness value
	Variation	–
	truncatedFreshnessValue	
	Type	SecOC_FreshnessArrayType
	Direction	OUT
	Comment	The truncated freshness value that has to be placed into the Secured-IPDU
	Variation	–
	truncatedFreshnessValueLength	
	Type	uint32
	Direction	INOUT
	Comment	The length in bits for the truncated freshness.
	Variation	–
Possible Errors	E_OK E_NOT_OK E_BUSY	

Operation	SPduTxConfirmation	
Comment	This operation is used by the SecOC to indicate that the Secured I-PDU has been initiated for transmission.	
Mapped to API	SecOC_SPduTxConfirmation	
Variation	–	
Parameters	freshnessValueId	
	Type	uint16
	Direction	IN
	Comment	Identifier of the freshness
	Variation	–
Possible Errors	E_OK	

⌋

8.8.3.3 Sending Default Authentication Information configuration service

[SWS_SecOC_00002] Definition of ClientServerInterface SendDefaultAuthenticationInformation

Upstream requirements: SRS_SecOC_00021

⌈

Name	SendDefaultAuthenticationInformation		
Comment	Sending Default Authentication Information configuration service.		
IsService	true		
Variation	({ecuc(SecOC/SecOCGeneral/SecOCDefaultAuthenticationInformationPattern.value != NULL)})		
Possible Errors	0	E_OK	Operation successful
	1	E_NOT_OK	Operation failed

Operation	SendDefaultAuthenticationInformation	
Comment	The service provides the ability to enable the sending of un-authenticated PDU to lower layer. (example: in case authentication build counter has reached the configuration value SecOCAuthenticationBuildAttempts or the query of the freshness function returns E_NOT_OK or the calculation of the authenticator has returned a non-recoverable error such as returning E_NOT_OK or KEY_FAILURE). This service is optional (the service is available only if SecOCDefaultAuthenticationInformationPattern is configured). If the service is not available or the service is available but the service was called with sendDefaultAuthenticationInformation as FALSE for a given FreshnessValueID, SecOC module shall remove the Authentic I-PDU from its internal buffer and cancel the transmission request in case the building of authentication Information failed. If the service is available and the service was called with sendDefaultAuthenticationInformation as TRUE for a given FreshnessValueID, SecOc will use SecOCDefaultAuthenticationInformationPattern as authentication Information and will not cancel the transmission request.	
Mapped to API	SecOC_SendDefaultAuthenticationInformation	
Variation	({ecuc(SecOC/SecOCRxPduProcessing/SecOCUseAuthDataFreshness)} == FALSE)	
Parameters	FreshnessValueID	
	Type	uint16
	Direction	IN
	Comment	ID of the Freshness Value for which sending SecOCDefaultAuthenticationInformationPattern should be enabled.
	Variation	–
	sendDefaultAuthenticationInformation	
	Type	boolean
	Direction	IN
	Comment	FALSE - sending SecOCDefaultAuthenticationInformationPattern shall be disabled for given FreshnessValueID TRUE - sending SecOCDefaultAuthenticationInformationPattern shall be enabled for given FreshnessValueID
	Variation	–
Possible Errors	E_OK E_NOT_OK	

⌋

8.8.3.4 Verification Status Provision Service

[SWS_SecOC_91016] Definition of ClientServerInterface VerificationStatusIndication

Upstream requirements: SRS_SecOC_00003, SRS_SecOC_00029

⌈

Name	VerificationStatusIndication		
Comment	This service realizes a notification service that is used to propagate the status of an authentication attempt from the SecOC module to an SW-C through RTE. This service can be configured such that: • Only "False" Verification Status is propagated to the application layer • Both "True" and "False" Verification Status are propagated to the application layer • No Verification Status is propagated to the application layer		
IsService	true		
Variation	–		
Possible Errors	0	E_OK	Operation successful
	1	E_NOT_OK	Operation failed

Operation	VerifyStatus	
Comment	This service provides the ability to inform the application about the result of the verification attempt of a received PDU by the SecOC module.	
Mapped to API	SecOC_VerifyStatus	
Variation	–	
Parameters	verificationStatus	
	Type	SecOC_VerificationStatusType
	Direction	IN
	Comment	The verificationStatus is a structure that provides details about the verification status and on which DataId and FreshnessValueId the verification was performed.
	Variation	–
Possible Errors	E_OK E_NOT_OK	

⌋

Note: The `VerificationStatusIndication` service is used to propagate the status of a verification attempt for a secured PDU from the SecOC module to an application software component. It can be used to continuously monitor the number of failed verification attempts and would allow setting up a security management system/intrusion detection system that is able to detect an attack flood and react with adequate dynamic countermeasures.

8.8.4 Implementation Data Types

8.8.4.1 SecOC_FreshnessArrayType

[SWS_SecOC_91012] Definition of ImplementationDataType SecOC_FreshnessArrayType

Upstream requirements: SRS_SecOC_00003, SRS_SecOC_00021, SRS_SecOC_00022

⌈

Name	SecOC_FreshnessArrayType		
Kind	Array	***Element type***	uint8
Size	SECOC_MAX_FRESHNESS_SIZE Elements		
Description	–		
Variation	–		
Available via	Rte_SecOC_Type.h		

⌋

8.8.4.2 SecOC_VerificationResultType

[SWS_SecOC_00149] Definition of ImplementationDataType SecOC_VerificationResultType

Upstream requirements: SRS_SecOC_00022

⌈

Name	SecOC_VerificationResultType		
Kind	Type		
Derived from	uint8		
Range	SECOC_ VERIFICATIONSUCCESS	0x00	Verification successful
	SECOC_ VERIFICATIONFAILURE	0x01	Verification not successful
	SECOC_ FRESHNESSFAILURE	0x02	Verification not successful because of wrong freshness value.
	SECOC_AUTHENTICA-TIONBUILDFAILURE	0x03	Verification not successful because of wrong build authentication codes
	SECOC_NO_ VERIFICATION	0x04	Verification has been skipped and the data has been provided to upper layer "as is". (only possible when SecOC_VerifyStatus Override is used)

▽

△

	SECOC_ VERIFICATIONFAILURE_ OVERWRITTEN	0x05	Verification failed, but the I-PDU was passed on to the upper layer due to the override status for this PDU. (only possible when Sec OC_VerifyStatusOverride is used)
Description	Enumeration to indicate verification results.		
Variation	–		
Available via	Rte_SecOC_Type.h		

⌋

8.8.4.3 SecOC_VerificationStatusType

[SWS_SecOC_00160] Definition of ImplementationDataType SecOC_Verification StatusType

Upstream requirements: SRS_SecOC_00022

⌈

Name	SecOC_VerificationStatusType	
Kind	Structure	
Elements	freshnessValueID	
	Type	uint16
	Comment	Identifier of the Freshness Value which resulted in the Verification Status
	verificationStatus	
	Type	SecOC_VerificationResultType
	Comment	Result of verification attempt: SECOC_VERIFICATIONSUCCESS = Verification successful SECOC_VERIFICATIONFAILURE = Verification not successful SECOC_FRESHNESSFAILURE = Verification not successful because of wrong freshness value SECOC_ AUTHENTICATIONBUILDFAILURE = Verification not successful because of wrong build authentication codes SECOC_NO_ VERIFICATION = No verification attempt was performed on this I-PDU and the I-PDU was passed on to the upper layer "as is". SECOC_ VERIFICATIONFAILURE_OVERWRITTEN = Verification failed, but the I-PDU was passed on to the upper layer due to the override status for this PDU.
	secOCDataId	
	Type	uint16
	Comment	Data ID of SecOCDataId
Description	Data structure to bundle the status of a verification attempt for a specific Freshness Value and Data ID	
Variation	–	
Available via	Rte_SecOC_Type.h	

⌋

8.8.4.4 SecOC_OverrideStatusType

[SWS_SecOC_00991] Definition of ImplementationDataType SecOC_OverrideStatusType

Upstream requirements: SRS_SecOC_00017

⌈

Name	SecOC_OverrideStatusType		
Kind	Type		
Derived from	uint8		
Range	SECOC_OVERRIDE_DROP_UNTIL_NOTICE	0x00	Until further notice, authenticator verification is not performed (no CSM call) I-PDU is dropped, verification result is set to SECOC_NO_VERIFICATION.
	SECOC_OVERRIDE_DROP_UNTIL_LIMIT	0x01	Until NumberOfMessagesTo Override is reached, authenticator verification is not performed (no CSM call) I-PDU is dropped, verification result is set to SECOC_NO_VERIFICATION.
	SECOC_OVERRIDE_CANCEL	0x02	Cancel Override of VerifyStatus.
	SECOC_OVERRIDE_PASS_UNTIL_NOTICE	0x40	Until further notice, authenticator verification is performed, I-PDU is sent to upper layer independent of verification result, verification result is set to SECOC_VERIFICATIONFAILURE_OVERWRITTEN in case of failed verification.
	SECOC_OVERRIDE_SKIP_UNTIL_LIMIT	0x41	Until NumberOfMessagesTo Override is reached, authenticator verification is not performed, I-PDU is sent to upper layer, verification result is set to SECOC_NO_VERIFICATION. If SecOCRxSecuredPduCollection is configured, SecOc shall process the SecOCRxAuthentic Pdu without waiting for SecOCRx CryptographicPdu.
	SECOC_OVERRIDE_PASS_UNTIL_LIMIT	0x42	Until NumberOfMessagesTo Override is reached, authenticator verification is performed, I-PDU is sent to upper layer independent of verification result, verification result is set to SECOC_VERIFICATIONFAILURE_OVERWRITTEN in case of failed verification.

▽

△

	SECOC_OVERRIDE_SKIP_UNTIL_NOTICE	0x43	Until further notice, authenticator verification is not performed, I-PDU is sent to upper layer, verification result is set to SECOC_NO_VERIFICATION. If SecOCRxSecuredPduCollection is configured, SecOc shall process the SecOCRxAuthentic Pdu without waiting for SecOCRx CryptographicPdu.
Description	Defines possibilities to override the verification status.		
Variation	–		
Available via	Rte_SecOC_Type.h		

⌋

8.8.5 Ports

8.8.5.1 Freshness Management

[SWS_SecOC_91001] Definition of Port FreshnessManagement required by module SecOC

Upstream requirements: SRS_SecOC_00003

⌈

Name	FreshnessManagement		
Kind	RequiredPort	***Interface***	FreshnessManagement
Description	Port for the provision of freshness for SecOC.		
Variation	({ecuc(SecOC/SecOCGeneral/SecOCQueryFreshnessValue)} == RTE)		

⌋

[SWS_SecOC_91020] Definition of Port SendDefaultAuthenticationInformation provided by module SecOC

Upstream requirements: SRS_SecOC_00003

⌈

Name	SendDefaultAuthenticationInformation		
Kind	ProvidedPort	***Interface***	SendDefaultAuthenticationInformation
Description	–		
Variation	({ecuc(SecOC/SecOCGeneral/SecOCDefaultAuthenticationInformationPattern.value != NULL)})		

⌋

[SWS_SecOC_91021] Definition of Port VerificationStatus provided by module SecOC

Upstream requirements: SRS_SecOC_00003, SRS_SecOC_00029

⌈

Name	VerificationStatus		
Kind	ProvidedPort	***Interface***	VerificationStatus
Description	–		
Variation	–		

⌋

[SWS_SecOC_91022] Definition of Port VerifyStatusConfiguration provided by module SecOC

Upstream requirements: SRS_SecOC_00003, SRS_SecOC_00029

⌈

Name	VerifyStatusConfiguration		
Kind	ProvidedPort	***Interface***	VerifyStatusConfiguration
Description	–		
Variation	–		

⌋

[SWS_SecOC_91015] Definition of Port VerificationStatusNotification required by module SecOC

Upstream requirements: SRS_SecOC_00003, SRS_SecOC_00029

⌈

Name	VerificationStatusNotification		
Kind	RequiredPort	***Interface***	VerificationStatusIndication
Description	Port definition for the notification of the verification status for a client-Server interface.		
Variation	–		

⌋

Note: Only one port is provided for the verification status. Hence, only one SW-C is able to receive and process the status with this client-server interface.

9 Sequence diagrams

The sequence diagrams in the following sections show interactions between the SecOC module, the PduR and the upper layer and lower layer communication modules. These sequences serve as examples to express the different kinds of interactions that are served by the SecOC module for authentication and verification.

Note: The examples show the interaction with distinct bus interface (e.g FrIf), transport protocol module (e.g. CanTp) or upper layer communication module (e.g. COM) only. However, they are valid for other bus interfaces, transport protocol modules and upper layer communication modules as well.

9.1 Authentication of outgoing PDUs

9.1.1 Authentication during direct transmission

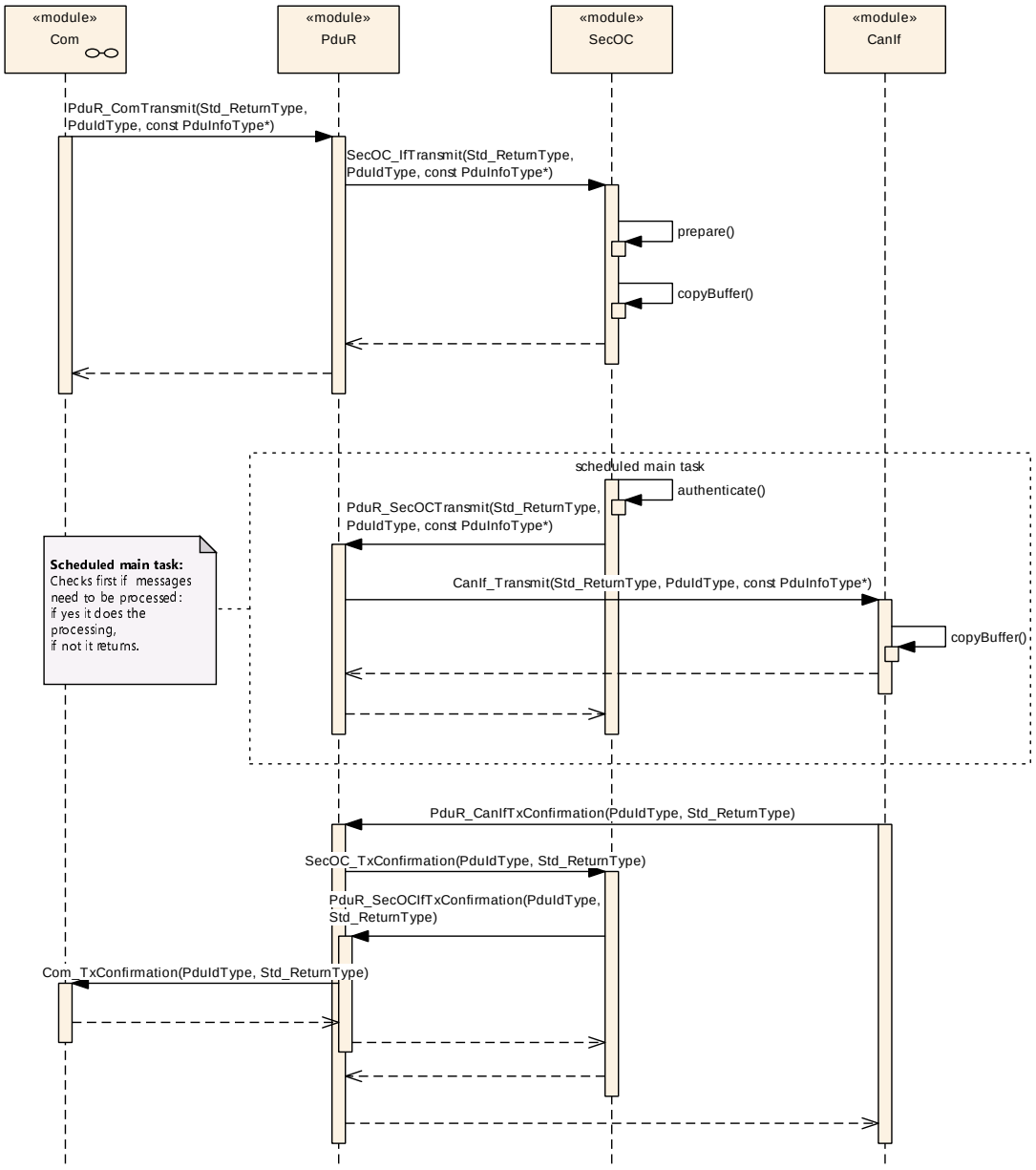

Figure 9.1: Authentication during direct transmission

9.1.2 Authentication during triggered transmission

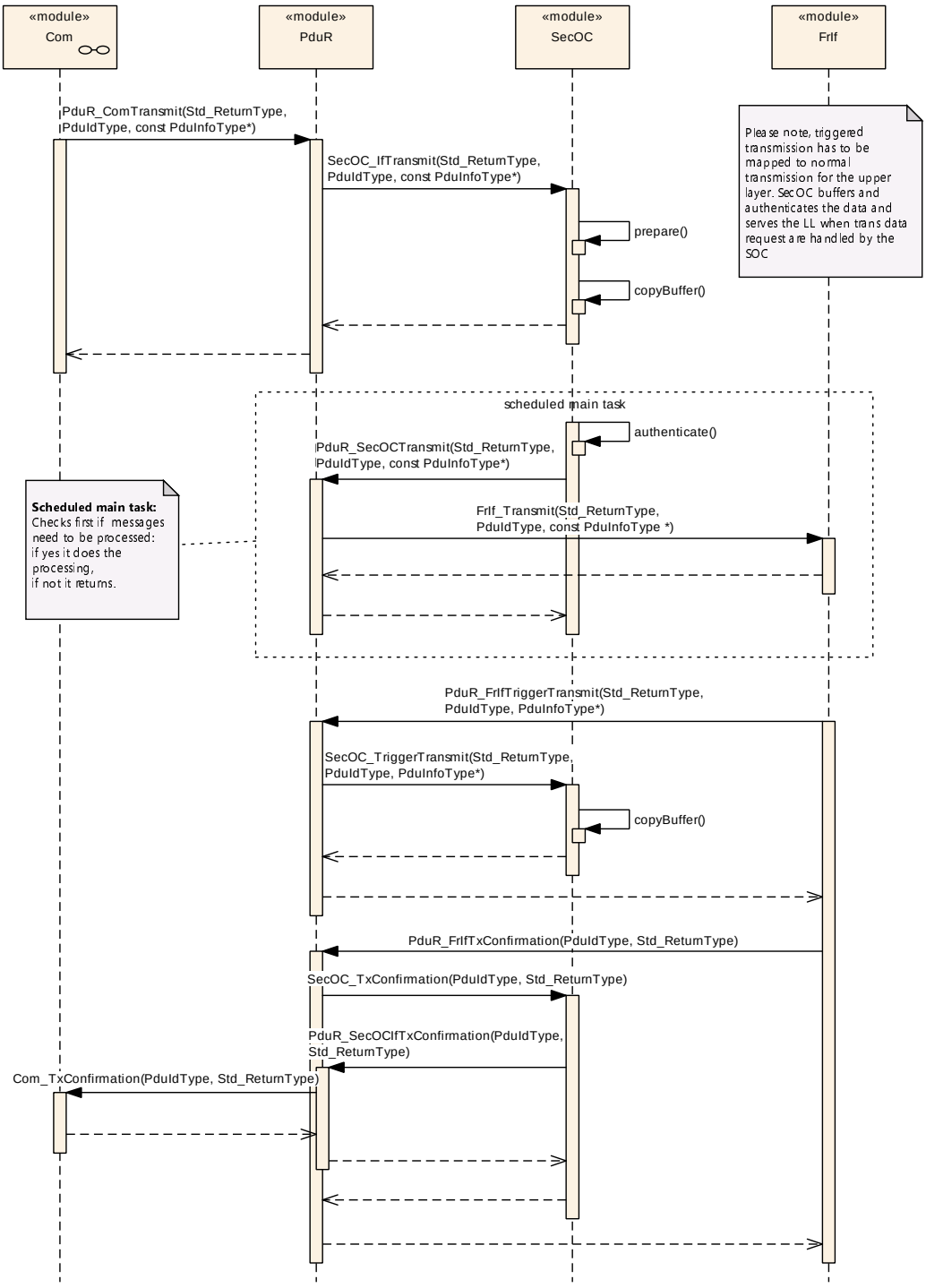

Figure 9.2: Authentication during Triggered Transmission

9.1.3 Authentication during transport protocol transmission

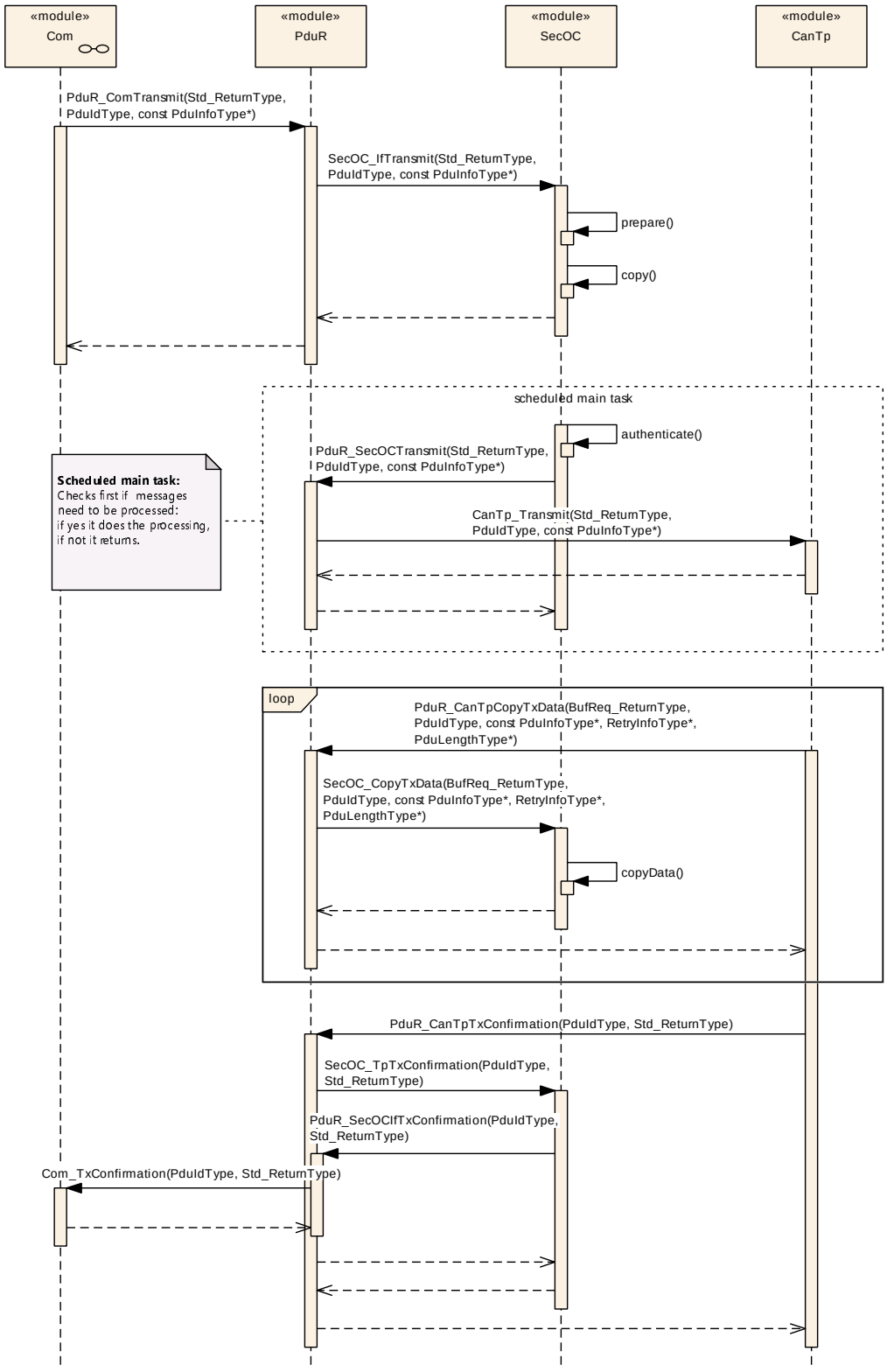

Figure 9.3: Authentication during TP transmission

9.1.4 Authentication with upper layer transport protocol

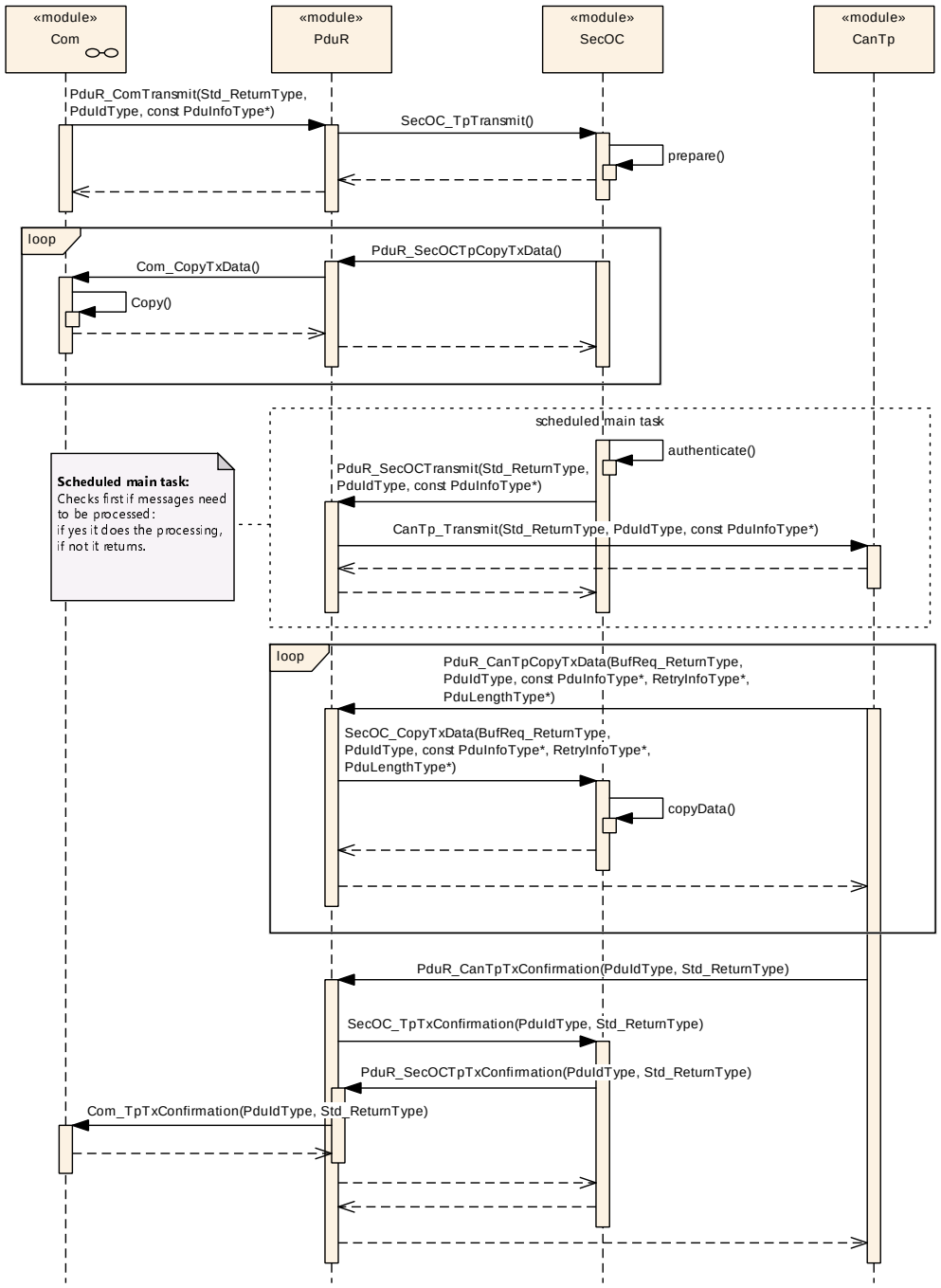

Figure 9.4: Authentication with upper layer TP

9.2 Verification of incoming PDUs

9.2.1 Verification during direct reception

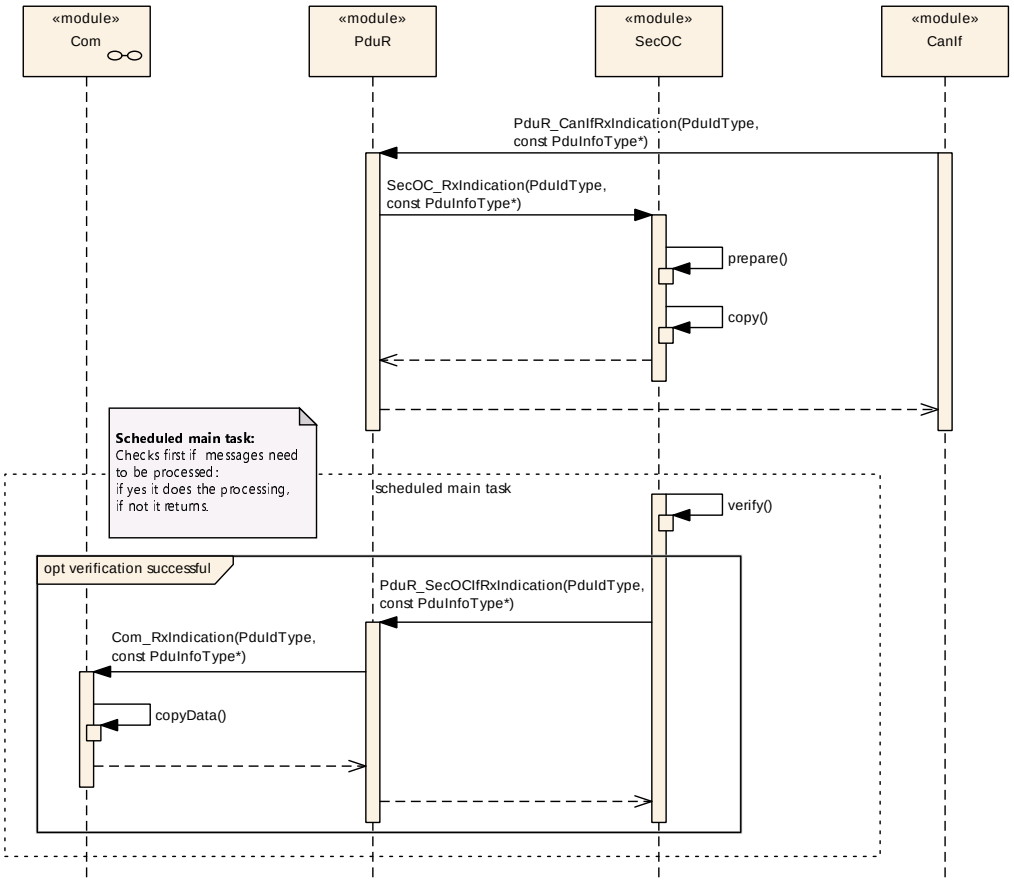

Figure 9.5: Verification during direct reception

9.2.2 Verification during transport protocol reception

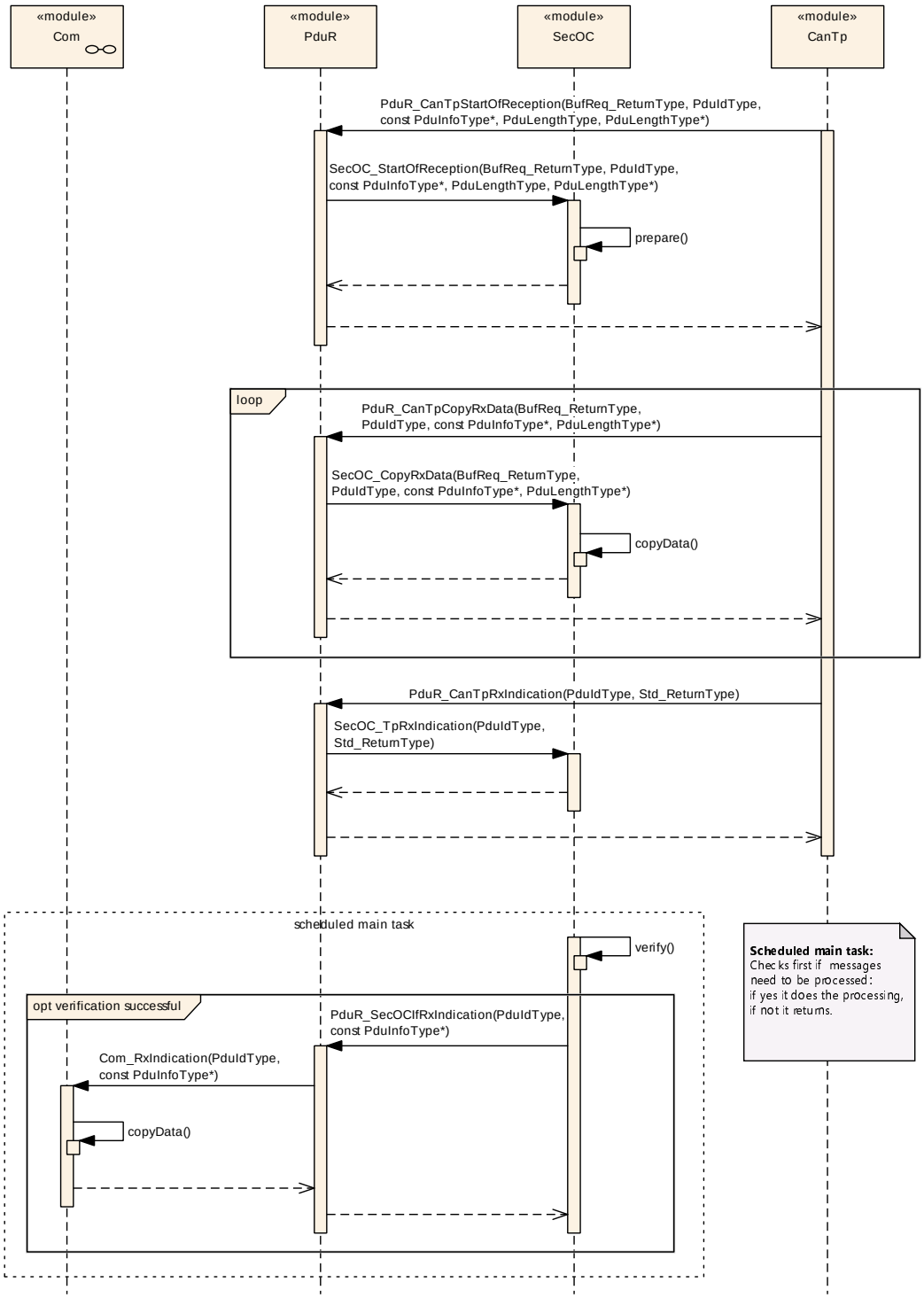

Figure 9.6: Verification during transport protocol reception

9.2.3 Verification with upper layer transport protocol

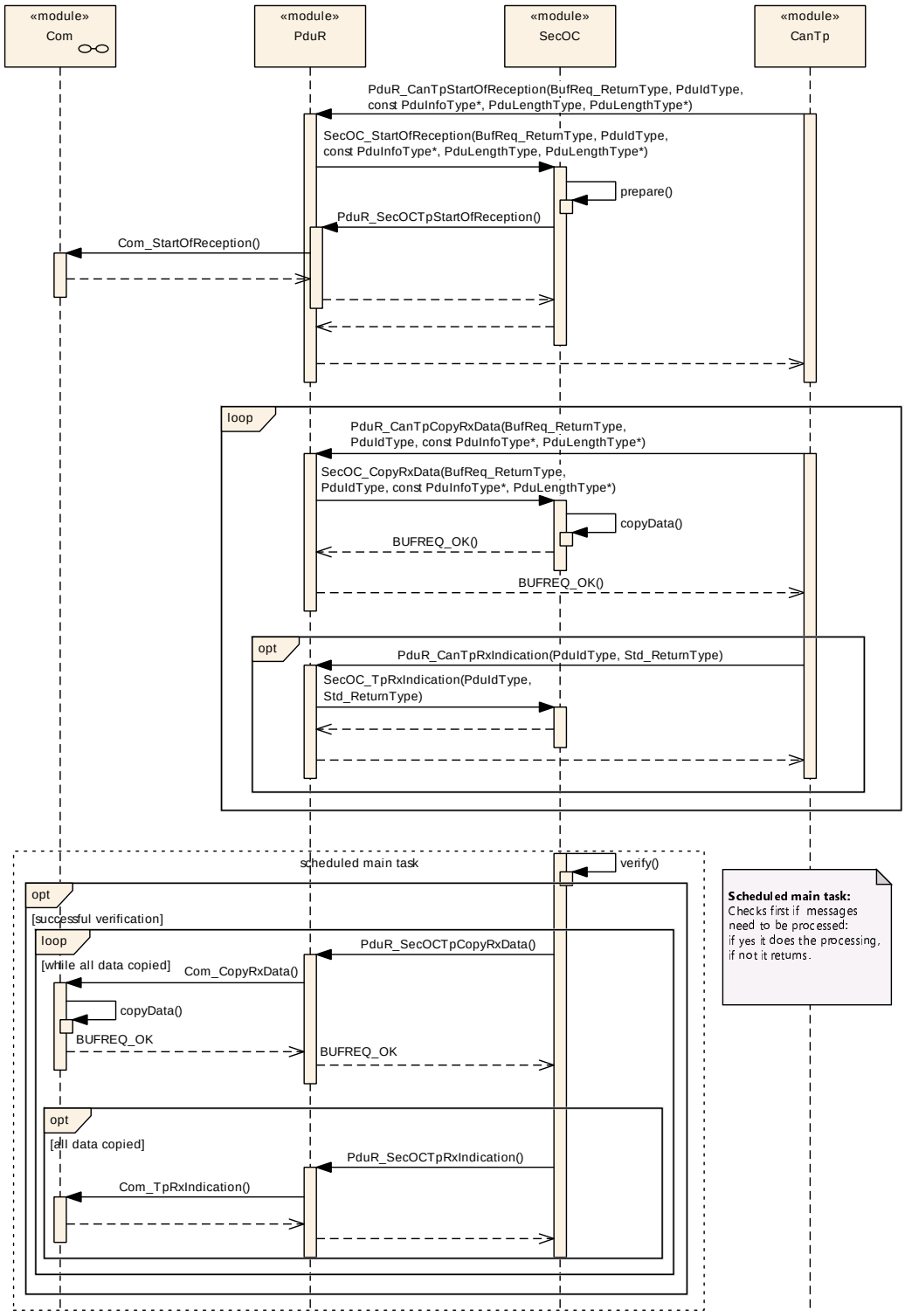

Figure 9.7: Verification with upper layer TP

9.3 Re-authentication Gateway

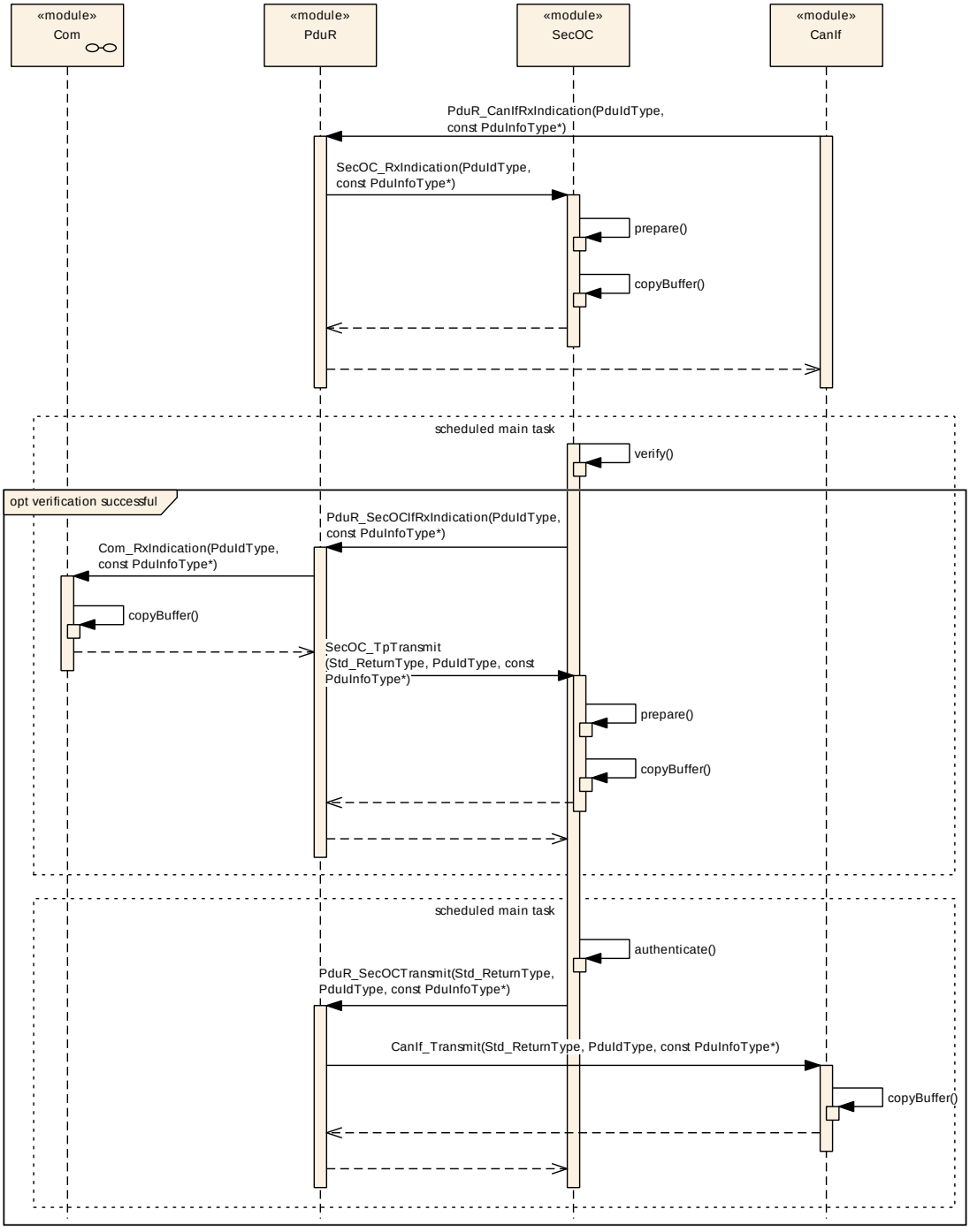

Figure 9.8: Verification and authentication in a gateway situation

9.4 Freshness Handling

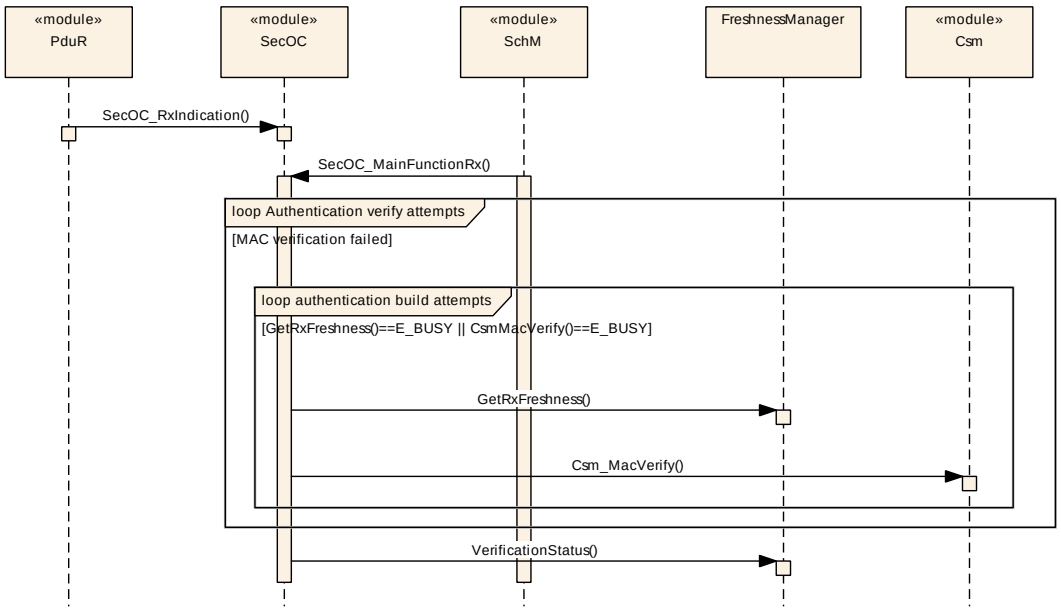

Figure 9.9: Freshness Handling

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 SecOC.

Chapter 10.3 specifies published information of the module SecOC.

10.1 How to read this chapter

For details refer to the chapter 10.1 “Introduction to configuration specification” in [4].

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.

For an overview of the AUTOSAR SecOC module’s configuration, see

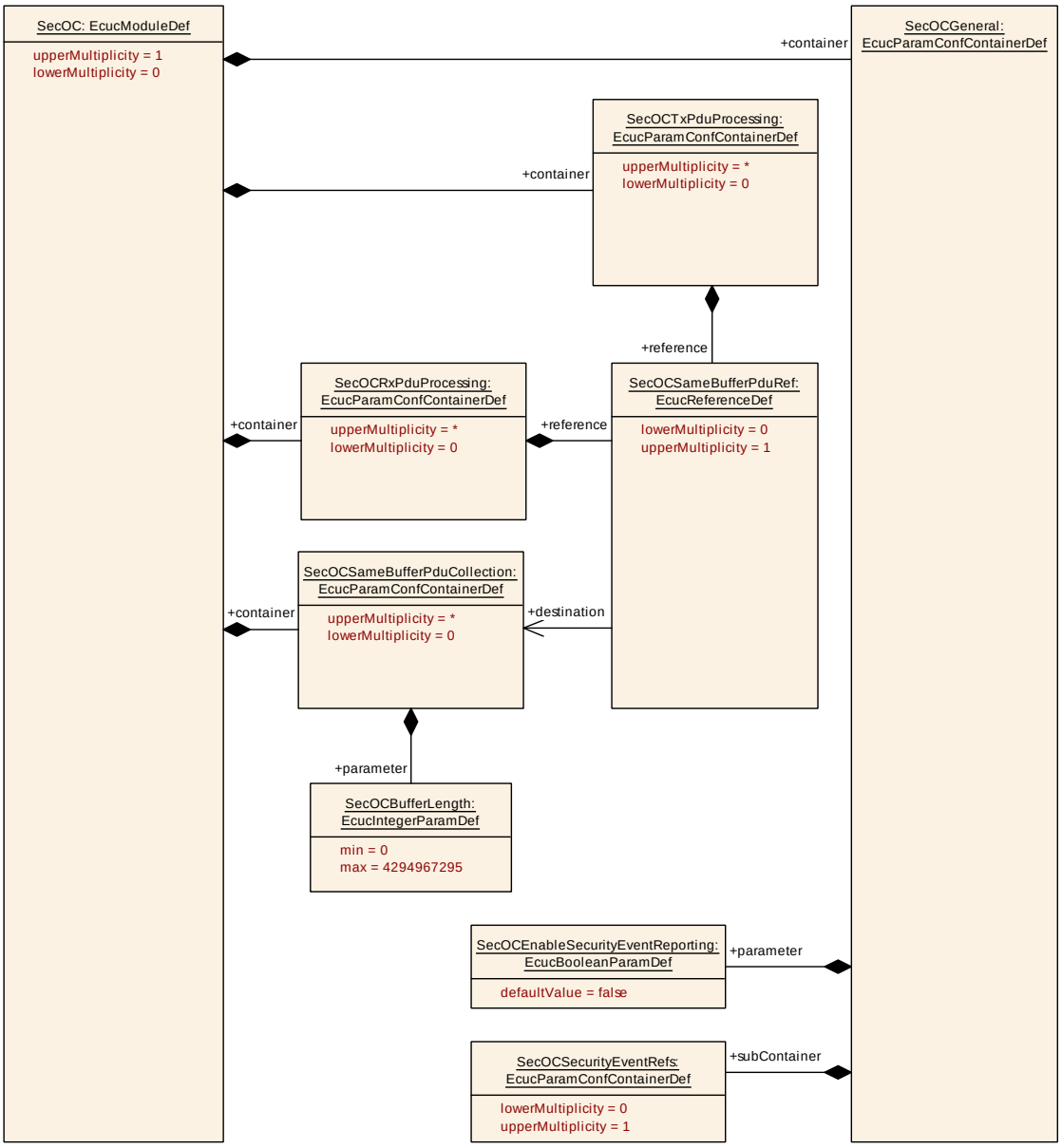

Figure 10.1: The AUTOSAR SecOC module's Configuration Overview

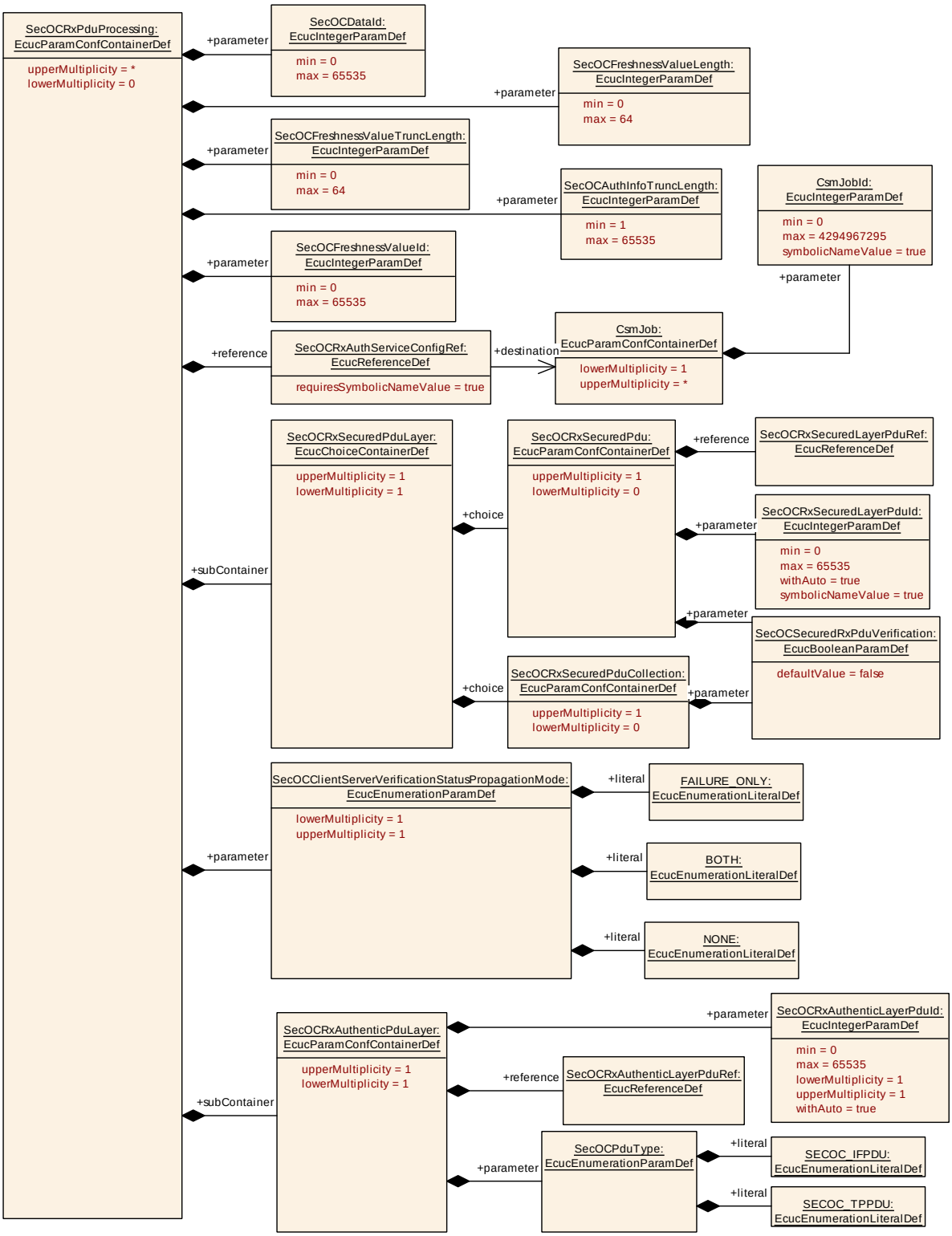

Figure 10.2: The AUTOSAR SecOC Rx Pdu Configuration Part 1

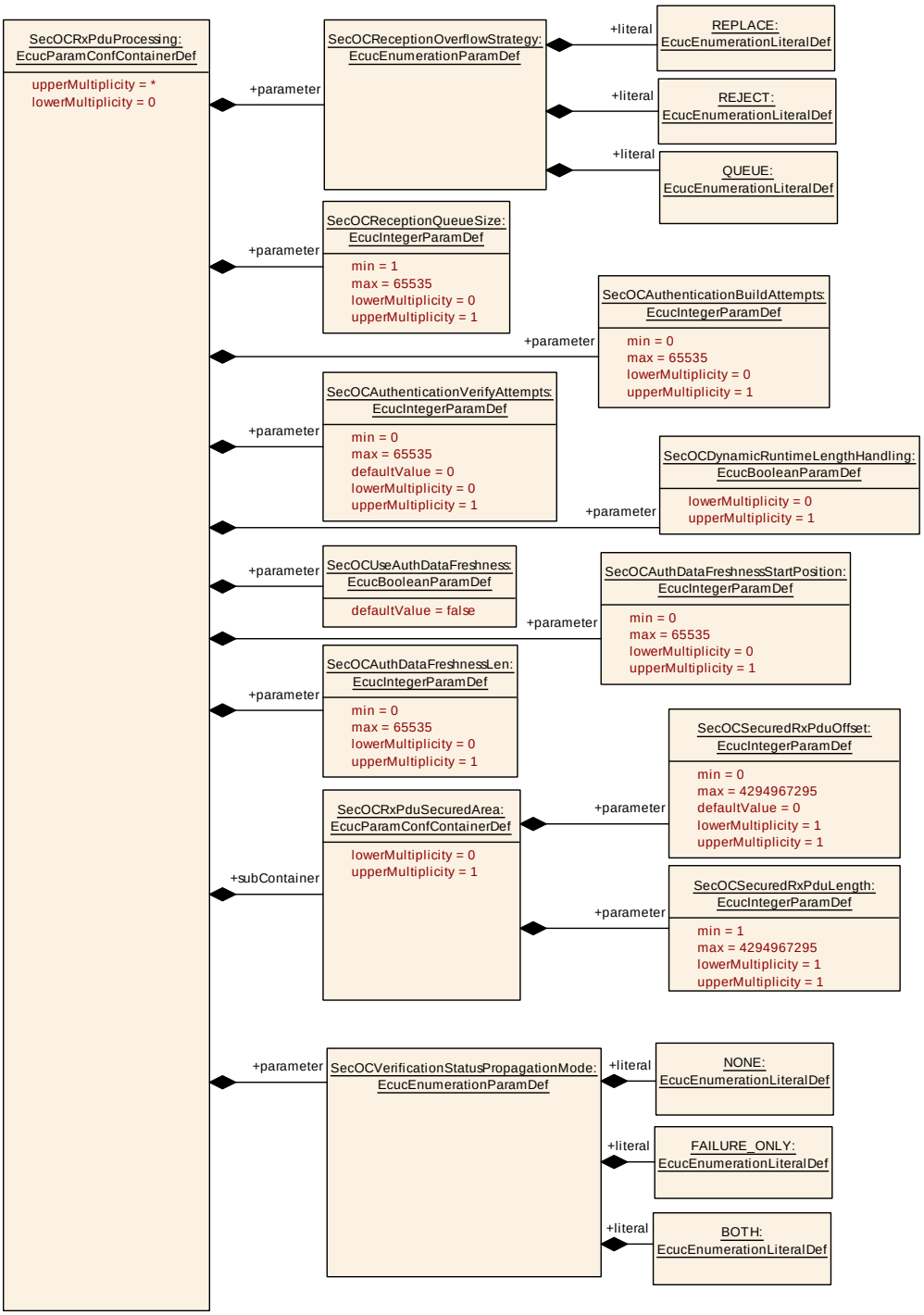

Figure 10.3: The AUTOSAR SecOC Rx Pdu Configuration Part 2

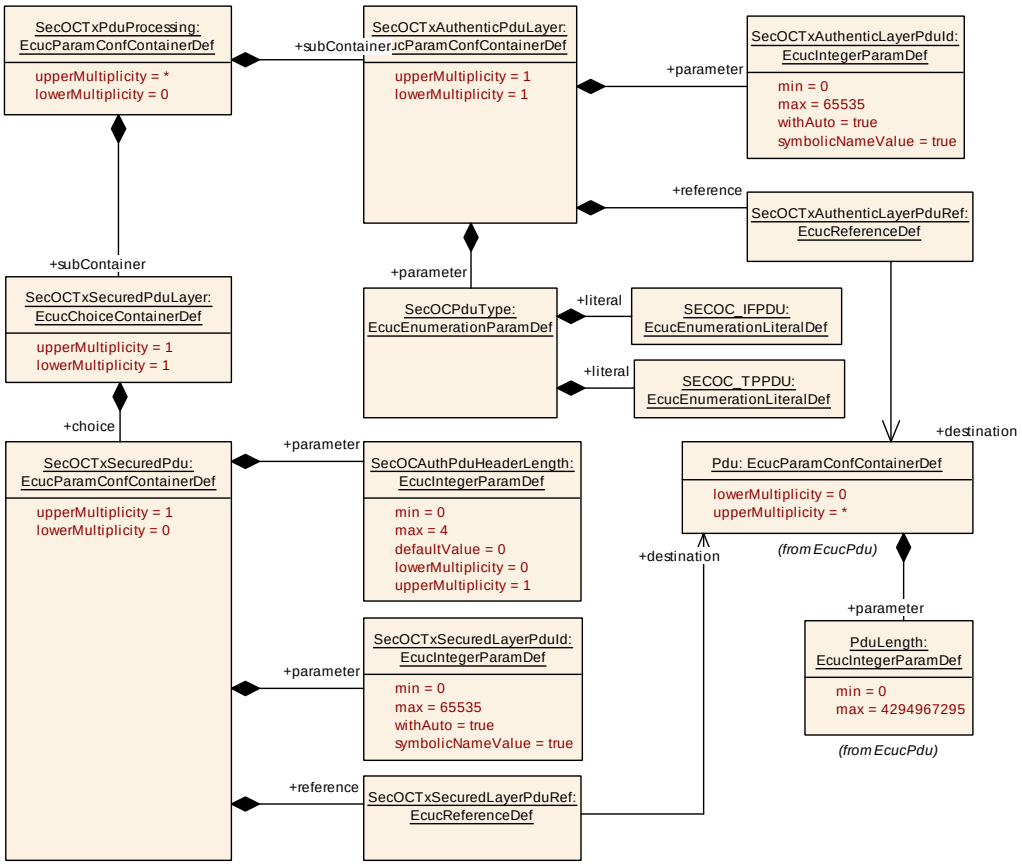

Figure 10.4: The AUTOSAR SecOC Tx Pdu Configuration Part 1

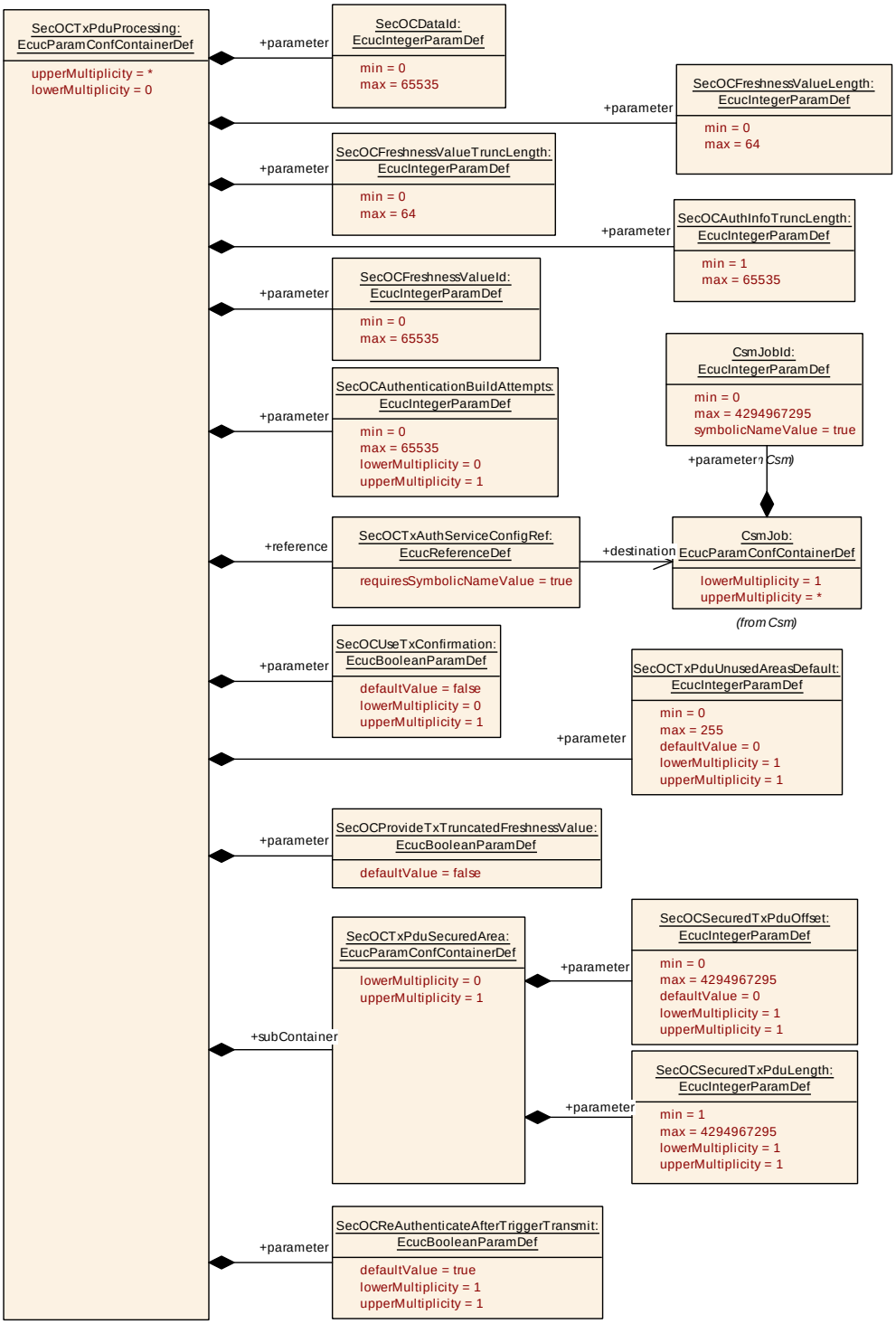

Figure 10.5: The AUTOSAR SecOC Tx Pdu Configuration Part 2

10.2.1 SecOC

[ECUC_SecOC_00001] Definition of EcucModuleDef SecOC ⌈

Module Name	SecOC
Description	Configuration of the SecOC (SecureOnboardCommunication) module.
Post-Build Variant Support	true
Supported Config Variants	VARIANT-LINK-TIME, VARIANT-POST-BUILD, VARIANT-PRE-COMPILE

Included Containers		
Container Name	Multiplicity	Scope / Dependency
SecOCGeneral	1	Contains the general configuration parameters of the SecOC module.
SecOCMainFunctionRx	0..*	Each element of this container defines one instance of SecOC_MainFunctionRx.
SecOCMainFunctionTx	0..*	Each element of this container defines one instance of SecOC_MainFunctionTx.
SecOCRxPduProcessing	0..*	Contains the parameters to configure the RxPdus to be verified by the SecOC module.
SecOCSameBufferPduCollection	0..*	SecOCBuffer configuration that may be used by a collection of Pdus.
SecOCTxPduProcessing	0..*	Contains the parameters to configure the TxPdus to be secured by the SecOC module.

⌋

10.2.2 SecOCGeneral

[ECUC_SecOC_00002] Definition of EcucParamConfContainerDef SecOCGeneral ⌈

Container Name	SecOCGeneral
Parent Container	SecOC
Description	Contains the general configuration parameters of the SecOC module.
Configuration Parameters	

Included Parameters		
Parameter Name	Multiplicity	ECUC ID
SecOCDefaultAuthenticationInformationPattern	0..1	[ECUC_SecOC_00098]
SecOCDevErrorDetect	1	[ECUC_SecOC_00007]
SecOCEnableForcedPassOverride	1	[ECUC_SecOC_00051]
SecOCEnableSecurityEventReporting	1	[ECUC_SecOC_00114]
SecOCIgnoreVerificationResult	1	[ECUC_SecOC_00052]
SecOCMaxAlignScalarType	1	[ECUC_SecOC_00047]
SecOCMaxTransmitRetries	1	[ECUC_SecOC_00119]
SecOCOverrideStatusWithDataId	0..1	[ECUC_SecOC_00099]
SecOCPropagateOnlyFinalVerificationStatus	1	[ECUC_SecOC_00112]
SecOCQueryFreshnessValue	1	[ECUC_SecOC_00078]
SecOCVerificationStatusCallout	0..*	[ECUC_SecOC_00004]
SecOCVersionInfoApi	1	[ECUC_SecOC_00003]

Included Containers		
Container Name	**Multiplicity**	**Scope / Dependency**
SecOCSecurityEventRefs	0..1	Container for the references to IdsMEvent elements representing the security events that the SecOC module shall report to the Ids M in case the coresponding security related event occurs (and if SecOCEnableSecurityEventReporting is set to "true"). The standardized security events in this container can be extended by vendor-specific security events. **Tags:** atp.Status=draft

⌋

[ECUC_SecOC_00098] Definition of EcucIntegerParamDef SecOCDefaultAuthenticationInformationPattern ⌈

Parameter Name	SecOCDefaultAuthenticationInformationPattern		
Parent Container	SecOCGeneral		
Description	The parameter describes the behaviour of SecOC when authentication build counter has reached the configuration value SecOCAuthenticationBuildAttempts, or the query of the freshness function returns E_NOT_OK or the calculation of the authenticator has returned a non-recoverable error such as returning E_NOT_OK or KEY_FAILURE. If the configuration parameter is not present, SecOC module shall remove the Authentic I-PDU from its internal buffer and cancel the transmission request If the configuration parameter is present, SecOC will use this value for each byte of Freshness Value and Authenticator when building the Authentication Information, and will not cancel the transmission request.		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	0 .. 255		
Default value	–		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

⌋

[ECUC_SecOC_00007] Definition of EcucBooleanParamDef SecOCDevErrorDetect ⌈

Parameter Name	SecOCDevErrorDetect
Parent Container	SecOCGeneral
Description	Switches the development error detection and notification on or off. • true: detection and notification is enabled. • false: detection and notification is disabled.

▽

△

Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

⌋

[ECUC_SecOC_00051] Definition of EcucBooleanParamDef SecOCEnableForcedPassOverride ⌈

Parameter Name	SecOCEnableForcedPassOverride		
Parent Container	SecOCGeneral		
Description	When this configuration option is set to TRUE then the functionality inside the function SecOC_VerifyStatusOverride to send I-PDUs to upper layer independent of the verification result is enabled.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

⌋

[ECUC_SecOC_00114] Definition of EcucBooleanParamDef SecOCEnableSecurityEventReporting

Status: DRAFT

⌈

Parameter Name	SecOCEnableSecurityEventReporting		
Parent Container	SecOCGeneral		
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		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	

▽

△

	Post-build time	–	
Scope / Dependency	scope: ECU		

⌋

[ECUC_SecOC_00052] Definition of EcucBooleanParamDef SecOCIgnoreVerificationResult ⌈

Parameter Name	SecOCIgnoreVerificationResult		
Parent Container	SecOCGeneral		
Description	The result of the authentication process (e.g. MAC Verify) is ignored after the first try and the SecOC proceeds like the result was a success. The calculation of the authenticator is still done, only its result will be ignored. • true: enabled (verification result is ignored). • false: disabled (verification result is NOT ignored).		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

⌋

[ECUC_SecOC_00047] Definition of EcucStringParamDef SecOCMaxAlignScalarType ⌈

Parameter Name	SecOCMaxAlignScalarType		
Parent Container	SecOCGeneral		
Description	The scalar type which has the maximum alignment restrictions on the given platform. This type can be e.g. uint8, uint16 or uint32.		
Multiplicity	1		
Type	EcucStringParamDef		
Default value	–		
Regular Expression	–		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

⌋

[ECUC_SecOC_00119] Definition of EcucIntegerParamDef SecOCMaxTransmitRetries ⌈

Parameter Name	SecOCMaxTransmitRetries		
Parent Container	SecOCGeneral		
Description	Maximum number of retries to send a secured I-PDU in case PduR_SecOCTransmit returns E_NOT_OK.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 255		
Default value	10		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

⌋

[ECUC_SecOC_00099] Definition of EcucBooleanParamDef SecOCOverrideStatusWithDataId ⌈

Parameter Name	SecOCOverrideStatusWithDataId		
Parent Container	SecOCGeneral		
Description	This option defines if the parameter "ValueId" of the function SecOC_VerifyStatusOverride() accepts the freshness value (as a collection of one or more Secured I-PDUs to freshness) or the dataID for individual Secured I-PDUs. • true: Function SecOC_VerifyStatusOverride accepts SecOCDataId as parameter. • false: Function SecOC_VerifyStatusOverride accepts SecOCFreshnessValueId as parameter.		
Multiplicity	0..1		
Type	EcucBooleanParamDef		
Default value	false		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

⌋

[ECUC_SecOC_00112] Definition of EcucBooleanParamDef SecOCPropagate OnlyFinalVerificationStatus ⌈

Parameter Name	SecOCPropagateOnlyFinalVerificationStatus		
Parent Container	SecOCGeneral		
Description	This parameter Is used to specify if the verification status shall be reported only after the final determination of the verification status (TRUE) or on every verification attempt (FALSE).		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

⌋

[ECUC_SecOC_00078] Definition of EcucEnumerationParamDef SecOCQuery FreshnessValue ⌈

Parameter Name	SecOCQueryFreshnessValue		
Parent Container	SecOCGeneral		
Description	This parameter specifies if the freshness value shall be determined through a C-function (CD) or a software component (SW-C).		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	CFUNC	The SecOC queries the freshness for every PDU to process using C function API	
	RTE	The SecOC queries the freshness for every PDU to process using the Rte service port Freshness Management	
Default value	CFUNC		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency			

⌋

[ECUC_SecOC_00004] Definition of EcucFunctionNameDef SecOCVerificationStatusCallout ⌈

Parameter Name	SecOCVerificationStatusCallout		
Parent Container	SecOCGeneral		
Description	Entry address of the customer specific call out routine which shall be invoked in case of a verification attempt.		
Multiplicity	0..*		
Type	EcucFunctionNameDef		
Default value	–		
Regular Expression	–		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

⌋

[ECUC_SecOC_00003] Definition of EcucBooleanParamDef SecOCVersionInfoApi ⌈

Parameter Name	SecOCVersionInfoApi		
Parent Container	SecOCGeneral		
Description	If true the SecOC_GetVersionInfo API is available.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

⌋

10.2.3 SecOCSecurityEventRefs

[ECUC_SecOC_00115] Definition of EcucParamConfContainerDef SecOCSecurityEventRefs

Status: DRAFT

⌈

Container Name	SecOCSecurityEventRefs		
Parent Container	SecOCGeneral		
Description	Container for the references to IdsMEvent elements representing the security events that the SecOC module shall report to the IdsM in case the coresponding security related event occurs (and if SecOCEnableSecurityEventReporting is set to "true"). The standardized security events in this container can be extended by vendor-specific security events. **Tags:** atp.Status=draft		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Configuration Parameters			

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
SEV_SECOC_FRESHNESS_NOT_AVAILABLE	0..1	[ECUC_SecOC_00117]
SEV_SECOC_MAC_VERIFICATION_FAILED	0..1	[ECUC_SecOC_00116]

No Included Containers

⌋

[ECUC_SecOC_00117] Definition of EcucReferenceDef SEV_SECOC_FRESHNESS_NOT_AVAILABLE

Status: DRAFT

⌈

Parameter Name	SEV_SECOC_FRESHNESS_NOT_AVAILABLE		
Parent Container	SecOCSecurityEventRefs		
Description	Failed to get freshness value from FvM. **Tags:** atp.Status=draft		
Multiplicity	0..1		
Type	Symbolic name reference to IdsMEvent		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

⌋

[ECUC_SecOC_00116] Definition of EcucReferenceDef SEV_SECOC_MAC_VERIFICATION_FAILED

Status: DRAFT

⌈

Parameter Name	SEV_SECOC_MAC_VERIFICATION_FAILED		
Parent Container	SecOCSecurityEventRefs		
Description	MAC verification of a received PDU failed. **Tags:** atp.Status=draft		
Multiplicity	0..1		
Type	Symbolic name reference to IdsMEvent		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

⌋

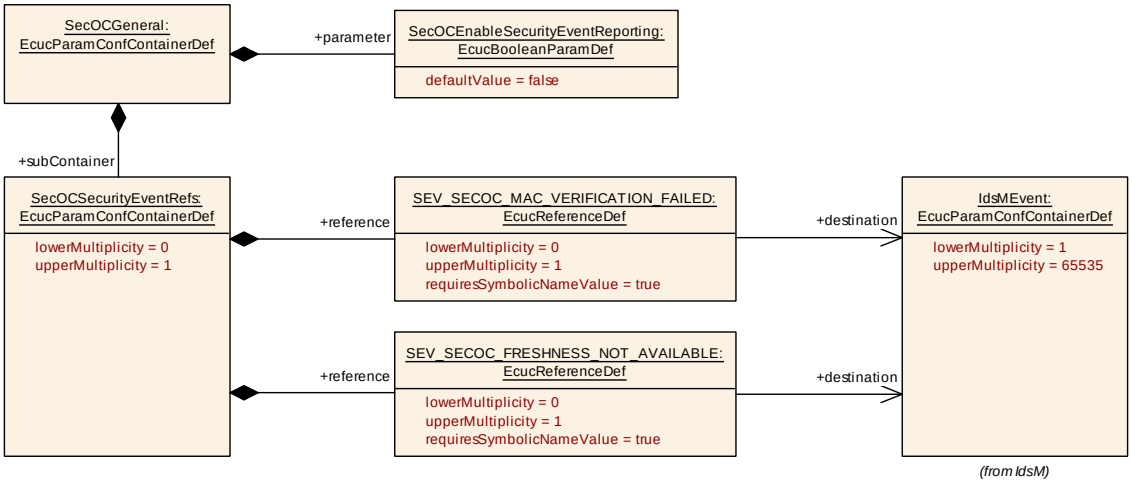

Figure 10.6: SecOCSecurityEventRefs configuration

10.2.4 SecOCMainFunctionRx

[ECUC_SecOC_00104] Definition of EcucParamConfContainerDef SecOCMainFunctionRx ⌈

Container Name	SecOCMainFunctionRx
Parent Container	SecOC
Description	Each element of this container defines one instance of SecOC_MainFunctionRx.
Configuration Parameters	

Included Parameters		
Parameter Name	Multiplicity	ECUC ID
SecOCMainFunctionPeriodRx	1	[ECUC_SecOC_00106]
SecOCMainFunctionRxPartitionRef	1	[ECUC_SecOC_00107]

No Included Containers

⌋

[ECUC_SecOC_00106] Definition of EcucFloatParamDef SecOCMainFunctionPeriodRx ⌈

Parameter Name	SecOCMainFunctionPeriodRx		
Parent Container	SecOCMainFunctionRx		
Description	Allows to configure the time for the respective MainFunction instance of the Rx path (as float in seconds).		
Multiplicity	1		
Type	EcucFloatParamDef		
Range	]0 .. INF[		
Default value	–		
Value Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

⌋

[ECUC_SecOC_00107] Definition of EcucReferenceDef SecOCMainFunctionRxPartitionRef ⌈

Parameter Name	SecOCMainFunctionRxPartitionRef		
Parent Container	SecOCMainFunctionRx		
Description	Reference to EcucPartition, where the according SecOC_MainFunction instance is assigned to.		
Multiplicity	1		
Type	Reference to EcucPartition		
Value Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

⌋

10.2.5 SecOCMainFunctionTx

[ECUC_SecOC_00105] Definition of EcucParamConfContainerDef SecOCMainFunctionTx ⌈

Container Name	SecOCMainFunctionTx
Parent Container	SecOC
Description	Each element of this container defines one instance of SecOC_MainFunctionTx.
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
SecOCMainFunctionPeriodTx	1	[ECUC_SecOC_00108]
SecOCMainFunctionTxPartitionRef	1	[ECUC_SecOC_00109]

No Included Containers

⌋

[ECUC_SecOC_00108] Definition of EcucFloatParamDef SecOCMainFunctionPeriodTx ⌈

Parameter Name	SecOCMainFunctionPeriodTx		
Parent Container	SecOCMainFunctionTx		
Description	Allows to configure the time for the respective MainFunction instance of the Tx path (as float in seconds).		
Multiplicity	1		
Type	EcucFloatParamDef		
Range	]0 .. INF[		
Default value	–		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

⌋

[ECUC_SecOC_00109] Definition of EcucReferenceDef SecOCMainFunctionTxPartitionRef ⌈

Parameter Name	SecOCMainFunctionTxPartitionRef
Parent Container	SecOCMainFunctionTx
Description	Reference to EcucPartition, where the according SecOC_MainFunction instance is assigned to.
Multiplicity	1
Type	Reference to EcucPartition

▽

△

Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

⌋

10.2.6 SecOCSameBufferPduCollection

[ECUC_SecOC_00009] Definition of EcucParamConfContainerDef SecOCSameBufferPduCollection ⌈

Container Name	SecOCSameBufferPduCollection		
Parent Container	SecOC		
Description	SecOCBuffer configuration that may be used by a collection of Pdus.		
Post-Build Variant Multiplicity	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Configuration Parameters			

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
SecOCBufferLength	1	[ECUC_SecOC_00008]

No Included Containers

⌋

[ECUC_SecOC_00008] Definition of EcucIntegerParamDef SecOCBufferLength ⌈

Parameter Name	SecOCBufferLength		
Parent Container	SecOCSameBufferPduCollection		
Description	This parameter defines the Buffer in bytes that is used by the SecOC module.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 4294967295		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	

▽

△

Scope / Dependency	scope: local

⌋

10.2.7 SecOCRxPduProcessing

[ECUC_SecOC_00011] Definition of EcucParamConfContainerDef SecOCRxPduProcessing ⌈

Container Name	SecOCRxPduProcessing
Parent Container	SecOC
Description	Contains the parameters to configure the RxPdus to be verified by the SecOC module.
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
SecOCAuthDataFreshnessLen	0..1	[ECUC_SecOC_00082]
SecOCAuthDataFreshnessStartPosition	0..1	[ECUC_SecOC_00081]
SecOCAuthenticationBuildAttempts	0..1	[ECUC_SecOC_00079]
SecOCAuthenticationVerifyAttempts	0..1	[ECUC_SecOC_00080]
SecOCAuthInfoTruncLength	1	[ECUC_SecOC_00095]
SecOCClientServerVerificationStatusPropagationMode	1	[ECUC_SecOC_00113]
SecOCDataId	1	[ECUC_SecOC_00014]
SecOCDynamicRuntimeLengthHandling	0..1	[ECUC_SecOC_00118]
SecOCFreshnessValueId	1	[ECUC_SecOC_00021]
SecOCFreshnessValueLength	1	[ECUC_SecOC_00015]
SecOCFreshnessValueTruncLength	1	[ECUC_SecOC_00094]
SecOCReceptionOverflowStrategy	1	[ECUC_SecOC_00076]
SecOCReceptionQueueSize	0..1	[ECUC_SecOC_00077]
SecOCUseAuthDataFreshness	1	[ECUC_SecOC_00083]
SecOCVerificationStatusPropagationMode	1	[ECUC_SecOC_00046]
SecOCRxAuthServiceConfigRef	1	[ECUC_SecOC_00048]
SecOCRxPduMainFunctionRef	0..1	[ECUC_SecOC_00110]
SecOCSameBufferPduRef	0..1	[ECUC_SecOC_00010]

Included Containers		
Container Name	**Multiplicity**	**Scope / Dependency**
SecOCRxAuthenticPduLayer	1	This container specifies the Pdu that is transmitted by the Sec OC module to the PduR after the Mac was verified.
SecOCRxPduSecuredArea	0..1	This container specifies an area in the Authentic I-Pdu that will be the input to the Authenticator verification algorithm. If this container does not exist in the configuration the complete Authentic I-Pdu will be the input to the Authenticator verification algorithm.
SecOCRxSecuredPduLayer	1	This container specifies the Pdu that is received by the SecOC module from the PduR. For this Pdu the Mac verification is provided.

⌋

[ECUC_SecOC_00082] Definition of EcucIntegerParamDef SecOCAuthDataFreshnessLen ⌈

Parameter Name	SecOCAuthDataFreshnessLen		
Parent Container	SecOCRxPduProcessing		
Description	The length of the external authentic PDU data in bits (uint16).		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	0 .. 65535		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

⌋

[ECUC_SecOC_00081] Definition of EcucIntegerParamDef SecOCAuthDataFreshnessStartPosition ⌈

Parameter Name	SecOCAuthDataFreshnessStartPosition		
Parent Container	SecOCRxPduProcessing		
Description	This value determines the start position in bits (uint16) of the Authentic PDU that shall be passed on to the Freshness SWC. The bit counting is done according to TPS_SYST_01068 and the bit ordering is done according to TPS_SYST_01069.		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	0 .. 65535		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	

▽

△

Scope / Dependency	scope: ECU

⌋

[ECUC_SecOC_00079] Definition of EcucIntegerParamDef SecOCAuthentication BuildAttempts ⌈

Parameter Name	SecOCAuthenticationBuildAttempts		
Parent Container	SecOCRxPduProcessing, SecOCTxPduProcessing		
Description	This parameter specifies the number of authentication build attempts.		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	0 .. 65535		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

⌋

[ECUC_SecOC_00080] Definition of EcucIntegerParamDef SecOCAuthentication VerifyAttempts ⌈

Parameter Name	SecOCAuthenticationVerifyAttempts		
Parent Container	SecOCRxPduProcessing		
Description	This parameter specifies the number of authentication verify attempts that are to be carried out when the verification of the authentication information failed for a given Secured I-PDU. If zero is set, then only one authentication verification attempt is done.		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	0 .. 65535		
Default value	0		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

⌋

[ECUC_SecOC_00095] Definition of EcucIntegerParamDef SecOCAuthInfoTruncLength ⌈

Parameter Name	SecOCAuthInfoTruncLength		
Parent Container	SecOCRxPduProcessing, SecOCTxPduProcessing		
Description	This parameter defines the length in bits of the authentication code to be included in the payload of the Secured I-PDU.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	1 .. 65535		
Default value	–		
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

⌋

[ECUC_SecOC_00113] Definition of EcucEnumerationParamDef SecOCClientServerVerificationStatusPropagationMode ⌈

Parameter Name	SecOCClientServerVerificationStatusPropagationMode		
Parent Container	SecOCRxPduProcessing		
Description	This parameter is used to determine the propagation of the verification status through the client/server interface to an SW-C.		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	BOTH	Both "TRUE" and "FALSE" AuthenticationStatus is propagated to SW-C	
	FAILURE_ONLY	Only "FALSE" Authentication Status is propagated to SW-C	
	NONE	No Authentication Status for this PDU is provided to SW-C	
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

⌋

[ECUC_SecOC_00014] Definition of EcucIntegerParamDef SecOCDataId ⌈

Parameter Name	SecOCDataId
Parent Container	SecOCRxPduProcessing, SecOCTxPduProcessing
Description	This parameter defines a unique numerical identifier for the Secured I-PDU.
Multiplicity	1

▽

△

Type	EcucIntegerParamDef		
Range	0 .. 65535		
Default value	–		
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

⌋

[ECUC_SecOC_00118] Definition of EcucBooleanParamDef SecOCDynamicRuntimeLengthHandling ⌈

Parameter Name	SecOCDynamicRuntimeLengthHandling		
Parent Container	SecOCRxPduProcessing		
Description	Defines whether the length information for handling this received Pdu is taken from the configuration or from the actually provided length information during runtime. true: SecuredIPdu length information is taken from the actually provided length information during runtime. false: SecuredIPdu length information is taken from parameter PduLength of the Pdu.		
Multiplicity	0..1		
Type	EcucBooleanParamDef		
Default value	–		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

⌋

[ECUC_SecOC_00021] Definition of EcucIntegerParamDef SecOCFreshnessValueId ⌈

Parameter Name	SecOCFreshnessValueId	
Parent Container	SecOCRxPduProcessing, SecOCTxPduProcessing	
Description	This parameter defines the Id of the Freshness Value. The Freshness Value might be a normal counter or a time value.	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	0 .. 65535	

▽

△

Default value	–		
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

⌋

[ECUC_SecOC_00015] Definition of EcucIntegerParamDef SecOCFreshness ValueLength ⌈

Parameter Name	SecOCFreshnessValueLength		
Parent Container	SecOCRxPduProcessing, SecOCTxPduProcessing		
Description	This parameter defines the complete length in bits of the Freshness Value. As long as the key doesn't change the counter shall not overflow. The length of the counter shall be determined based on the expected life time of the corresponding key and frequency of usage of the counter.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 64		
Default value	–		
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

⌋

[ECUC_SecOC_00094] Definition of EcucIntegerParamDef SecOCFreshness ValueTruncLength ⌈

Parameter Name	SecOCFreshnessValueTruncLength		
Parent Container	SecOCRxPduProcessing, SecOCTxPduProcessing		
Description	This parameter defines the length in bits of the Freshness Value to be included in the payload of the Secured I-PDU. This length is specific to the least significant bits of the complete Freshness Counter. If the parameter is 0 no Freshness Value is included in the Secured I-PDU.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 64		
Default value	–		
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD

▽

△

Scope / Dependency	scope: local dependency: SecOCFreshnessCounterTxLength <= SecOCFreshnessCounterLength

⌋

[ECUC_SecOC_00076] Definition of EcucEnumerationParamDef SecOCReceptionOverflowStrategy ⌈

Parameter Name	SecOCReceptionOverflowStrategy		
Parent Container	SecOCRxPduProcessing		
Description	This parameter defines the overflow strategy for receiving PDUs		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	QUEUE	Subsequent received message will be queued	
	REJECT	Subsequent received message will be discarded	
	REPLACE	Subsequent received message will replace the currently processed message	
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

⌋

[ECUC_SecOC_00077] Definition of EcucIntegerParamDef SecOCReception QueueSize ⌈

Parameter Name	SecOCReceptionQueueSize		
Parent Container	SecOCRxPduProcessing		
Description	This parameter defines the queue size in case the overflow strategy for receiving PDUs is set to QUEUE.		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	1 .. 65535		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

⌋

[ECUC_SecOC_00083] Definition of EcucBooleanParamDef SecOCUseAuthDataFreshness ⌈

Parameter Name	SecOCUseAuthDataFreshness		
Parent Container	SecOCRxPduProcessing		
Description	A Boolean value that indicates if a part of the Authentic-PDU shall be passed on to the SWC that verifies and generates the Freshness. If it is set to TRUE, the values SecOCAuthDataFreshnessStartPosition and SecOCAuthDataFreshnessLen must be set to specify the bit position and length within the Authentic-PDU.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

⌋

[ECUC_SecOC_00046] Definition of EcucEnumerationParamDef SecOCVerificationStatusPropagationMode ⌈

Parameter Name	SecOCVerificationStatusPropagationMode		
Parent Container	SecOCRxPduProcessing		
Description	This parameter is used to describe the propagation of the status of each verification attempt from the SecOC module to SWCs.		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	BOTH	Both "True" and "False" AuthenticationStatus is propagated to SWC	
	FAILURE_ONLY	Only "False" AuthenticationStatus is propagated to SWC	
	NONE	No AuthenticationStatus is propagated to SWC	
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

⌋

[ECUC_SecOC_00048] Definition of EcucReferenceDef SecOCRxAuthServiceConfigRef ⌈

Parameter Name	SecOCRxAuthServiceConfigRef
Parent Container	SecOCRxPduProcessing
Description	This reference is used to define which crypto service function is called for authentication. If PDUs with a dynamic length are used (e.g. CanTP or Dynamic Length PDUs) a MAC algorithm has to be chosen, that is not vulnerable to length extension attack (e.g. CMAC/HMAC).
Multiplicity	1
Type	Symbolic name reference to CsmJob
Post-Build Variant Value	false
Scope / Dependency	

⌋

[ECUC_SecOC_00110] Definition of EcucReferenceDef SecOCRxPduMainFunctionRef ⌈

Parameter Name	SecOCRxPduMainFunctionRef		
Parent Container	SecOCRxPduProcessing		
Description	Reference to the SecOC_MainFunctionRx this PDU belongs to. Mandatory, if multiple main functions are defined.		
Multiplicity	0..1		
Type	Reference to SecOCMainFunctionRx		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

⌋

[ECUC_SecOC_00010] Definition of EcucReferenceDef SecOCSameBufferPduRef ⌈

Parameter Name	SecOCSameBufferPduRef		
Parent Container	SecOCRxPduProcessing, SecOCTxPduProcessing		
Description	This reference is used to collect Pdus that are using the same SecOC buffer.		
Multiplicity	0..1		
Type	Reference to SecOCSameBufferPduCollection		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	

▽

△

Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

⌋

10.2.8 SecOCRxSecuredPduLayer

[ECUC_SecOC_00041] Definition of EcucChoiceContainerDef SecOCRxSecuredPduLayer ⌈

Choice Container Name	SecOCRxSecuredPduLayer
Parent Container	SecOCRxPduProcessing
Description	This container specifies the Pdu that is received by the SecOC module from the PduR. For this Pdu the Mac verification is provided.

No Included Parameters

Container Choices		
Container Name	**Multiplicity**	**Scope / Dependency**
SecOCRxSecuredPdu	0..1	This container specifies the Pdu that is received by the SecOC module from the PduR. For this Pdu the Mac verification is provided.
SecOCRxSecuredPduCollection	0..1	This container specifies two Pdus that are received by the Sec OC module from the PduR and a message linking between them. SecOCRxAuthenticPdu contains the original Authentic I-PDU, i.e. the secured data, and the SecOCRxCryptographicPdu contains the Authenticator, i.e. the actual Authentication Information.

⌋

10.2.9 SecOCRxSecuredPdu

[ECUC_SecOC_00069] Definition of EcucParamConfContainerDef SecOCRxSecuredPdu ⌈

Container Name	SecOCRxSecuredPdu
Parent Container	SecOCRxSecuredPduLayer
Description	This container specifies the Pdu that is received by the SecOC module from the PduR. For this Pdu the Mac verification is provided.
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
SecOCAuthPduHeaderLength	0..1	[ECUC_SecOC_00093]
SecOCRxSecuredLayerPduId	1	[ECUC_SecOC_00043]
SecOCSecuredRxPduVerification	1	[ECUC_SecOC_00092]
SecOCRxSecuredLayerPduRef	1	[ECUC_SecOC_00042]

No Included Containers

⌋

For parameter table [ECUC_SecOC_00093] SecOCAuthPduHeaderLength, see definition below container SecOCRxAuthenticPdu.

[ECUC_SecOC_00043] Definition of EcucIntegerParamDef SecOCRxSecuredLayerPduId ⌈

Parameter Name	SecOCRxSecuredLayerPduId		
Parent Container	SecOCRxSecuredPdu		
Description	PDU identifier assigned by SecOC module. Used by PduR for SecOC_[If|Tp]Rx Indication.		
Multiplicity	1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 65535		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local withAuto = true		

⌋

[ECUC_SecOC_00092] Definition of EcucBooleanParamDef SecOCSecuredRxPduVerification ⌈

Parameter Name	SecOCSecuredRxPduVerification		
Parent Container	SecOCRxSecuredPdu, SecOCRxSecuredPduCollection		
Description	This parameter defines whether the signature authentication or MAC verification shall be performed on this Secured I-PDU. If set to false, the SecOC module extracts the Authentic I-PDU from the Secured I-PDU without verification.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	All Variants

▽

△

	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

⌋

[ECUC_SecOC_00042] Definition of EcucReferenceDef SecOCRxSecuredLayerPduRef ⌈

Parameter Name	SecOCRxSecuredLayerPduRef		
Parent Container	SecOCRxSecuredPdu		
Description	Reference to the global Pdu.		
Multiplicity	1		
Type	Reference to Pdu		
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

⌋

10.2.10 SecOCRxAuthenticPduLayer

[ECUC_SecOC_00044] Definition of EcucParamConfContainerDef SecOCRxAuthenticPduLayer ⌈

Container Name	SecOCRxAuthenticPduLayer
Parent Container	SecOCRxPduProcessing
Description	This container specifies the Pdu that is transmitted by the SecOC module to the PduR after the Mac was verified.
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
SecOCPduType	1	[ECUC_SecOC_00075]
SecOCRxAuthenticLayerPduId	1	[ECUC_SecOC_00102]
SecOCRxAuthenticLayerPduRef	1	[ECUC_SecOC_00045]

No Included Containers

⌋

[ECUC_SecOC_00075] Definition of EcucEnumerationParamDef SecOCPduType ⌈

Parameter Name	SecOCPduType		
Parent Container	SecOCRxAuthenticPduLayer, SecOCTxAuthenticPduLayer		
Description	This parameter defines API Type to use for communication with PduR.		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	SECOC_IFPDU	SECOC_IFPDU Interface communication API	
	SECOC_TPPDU	SECOC_TPPDU Transport Protocol communication API	
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

⌋

[ECUC_SecOC_00102] Definition of EcucIntegerParamDef SecOCRxAuthentic LayerPduId ⌈

Parameter Name	SecOCRxAuthenticLayerPduId		
Parent Container	SecOCRxAuthenticPduLayer		
Description	PDU identifier assigned by SecOC module. Used by PduR for SecOC_TpCancel Receive.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 65535		
Default value	–		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local withAuto = true		

⌋

[ECUC_SecOC_00045] Definition of EcucReferenceDef SecOCRxAuthenticLayer PduRef ⌈

Parameter Name	SecOCRxAuthenticLayerPduRef
Parent Container	SecOCRxAuthenticPduLayer
Description	Reference to the global Pdu.
Multiplicity	1
Type	Reference to Pdu
Post-Build Variant Value	true

▽

△

Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

⌋

10.2.11 SecOCRxSecuredPduCollection

[ECUC_SecOC_00067] Definition of EcucParamConfContainerDef SecOCRxSecuredPduCollection ⌈

Container Name	SecOCRxSecuredPduCollection
Parent Container	SecOCRxSecuredPduLayer
Description	This container specifies two Pdus that are received by the SecOC module from the Pdu R and a message linking between them. SecOCRxAuthenticPdu contains the original Authentic I-PDU, i.e. the secured data, and the SecOCRxCryptographicPdu contains the Authenticator, i.e. the actual Authentication Information.
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
SecOCSecuredRxPduVerification	1	[ECUC_SecOC_00092]

Included Containers		
Container Name	**Multiplicity**	**Scope / Dependency**
SecOCRxAuthenticPdu	1	This container specifies the PDU (that is received by the SecOC module from the PduR) which contains the Secured I-PDU Header and the Authentic I-PDU.
SecOCRxCryptographicPdu	1	This container specifies the Cryptographic Pdu that is received by the SecOC module from the PduR.
SecOCUseMessageLink	0..1	SecOC links an Authentic I-PDU and Cryptographic I-PDU together by repeating a specific part (Message Linker) of the Authentic I-PDU in the Cryptographic I-PDU.

⌋

For parameter table [ECUC_SecOC_00092] SecOCSecuredRxPduVerification, see definition below container SecOCRxSecuredPdu.

[SWS_SecOC_CONSTR_00265] ⌈Within the same configured `SecOCRxPduProcessing`, if `SecOCReceptionOverflowStrategy` is set to `QUEUE`, then `SecOCRxSecuredPduCollection` shall have multiplicity of 0.⌋

10.2.12 SecOCRxCryptographicPdu

[ECUC_SecOC_00064] Definition of EcucParamConfContainerDef SecOCRxCryptographicPdu ⌈

Container Name	SecOCRxCryptographicPdu
Parent Container	SecOCRxSecuredPduCollection
Description	This container specifies the Cryptographic Pdu that is received by the SecOC module from the PduR.
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
SecOCRxCryptographicPduId	1	[ECUC_SecOC_00065]
SecOCRxCryptographicPduRef	1	[ECUC_SecOC_00066]

No Included Containers

⌋

[ECUC_SecOC_00065] Definition of EcucIntegerParamDef SecOCRxCryptographicPduId ⌈

Parameter Name	SecOCRxCryptographicPduId		
Parent Container	SecOCRxCryptographicPdu		
Description	PDU identifier of the Cryptographic I-PDU assigned by SecOC module. Used by PduR for SecOC_IfRxIndication.		
Multiplicity	1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 65535		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local withAuto = true		

⌋

[ECUC_SecOC_00066] Definition of EcucReferenceDef SecOCRxCryptographicPduRef ⌈

Parameter Name	SecOCRxCryptographicPduRef
Parent Container	SecOCRxCryptographicPdu
Description	Reference to the global Pdu.

▽

△

Multiplicity	1		
Type	Reference to Pdu		
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

⌋

10.2.13 SecOCRxAuthenticPdu

[ECUC_SecOC_00061] Definition of EcucParamConfContainerDef SecOCRxAuthenticPdu ⌈

Container Name	SecOCRxAuthenticPdu
Parent Container	SecOCRxSecuredPduCollection
Description	This container specifies the PDU (that is received by the SecOC module from the Pdu R) which contains the Secured I-PDU Header and the Authentic I-PDU.
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
SecOCAuthPduHeaderLength	0..1	[ECUC_SecOC_00093]
SecOCRxAuthenticPduId	1	[ECUC_SecOC_00062]
SecOCRxAuthenticPduRef	1	[ECUC_SecOC_00063]

No Included Containers

⌋

[ECUC_SecOC_00093] Definition of EcucIntegerParamDef SecOCAuthPduHeaderLength ⌈

Parameter Name	SecOCAuthPduHeaderLength	
Parent Container	SecOCRxAuthenticPdu, SecOCRxSecuredPdu, SecOCTxAuthenticPdu, SecOCTxSecuredPdu	
Description	This parameter indicates the length (in bytes) of the Secured I-PDU Header in the Secured I-PDU. The length of zero means there's no header in the PDU.	
Multiplicity	0..1	
Type	EcucIntegerParamDef	
Range	0 .. 4	
Default value	0	

▽

△

Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

⌋

[ECUC_SecOC_00062] Definition of EcucIntegerParamDef SecOCRxAuthenticPduId ⌈

Parameter Name	SecOCRxAuthenticPduId		
Parent Container	SecOCRxAuthenticPdu		
Description	PDU identifier of the Authentic I-PDU assigned by SecOC module. Used by PduR for SecOC_IfRxIndication.		
Multiplicity	1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 65535		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local withAuto = true		

⌋

[ECUC_SecOC_00063] Definition of EcucReferenceDef SecOCRxAuthenticPduRef ⌈

Parameter Name	SecOCRxAuthenticPduRef		
Parent Container	SecOCRxAuthenticPdu		
Description	Reference to the global Pdu.		
Multiplicity	1		
Type	Reference to Pdu		
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

⌋

10.2.14 SecOCTxPduProcessing

[ECUC_SecOC_00012] Definition of EcucParamConfContainerDef SecOCTxPduProcessing ⌈

Container Name	SecOCTxPduProcessing
Parent Container	SecOC
Description	Contains the parameters to configure the TxPdus to be secured by the SecOC module.
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
SecOCAuthenticationBuildAttempts	0..1	[ECUC_SecOC_00079]
SecOCAuthInfoTruncLength	1	[ECUC_SecOC_00095]
SecOCDataId	1	[ECUC_SecOC_00014]
SecOCFreshnessValueId	1	[ECUC_SecOC_00021]
SecOCFreshnessValueLength	1	[ECUC_SecOC_00015]
SecOCFreshnessValueTruncLength	1	[ECUC_SecOC_00094]
SecOCProvideTxTruncatedFreshnessValue	1	[ECUC_SecOC_00084]
SecOCReAuthenticateAfterTriggerTransmit	1	[ECUC_SecOC_00103]
SecOCTxPduUnusedAreasDefault	1	[ECUC_SecOC_00101]
SecOCUseTxConfirmation	0..1	[ECUC_SecOC_00085]
SecOCSameBufferPduRef	0..1	[ECUC_SecOC_00010]
SecOCTxAuthServiceConfigRef	1	[ECUC_SecOC_00013]
SecOCTxPduMainFunctionRef	0..1	[ECUC_SecOC_00111]

Included Containers		
Container Name	**Multiplicity**	**Scope / Dependency**
SecOCTxAuthenticPduLayer	1	This container specifies the Pdu that is received by the SecOC module from the PduR. For this Pdu the Mac generation is provided.
SecOCTxPduSecuredArea	0..1	This container specifies an area in the Authentic I-Pdu that will be the input to the Authenticator generation algorithm. If this container does not exist in the configuration the complete Authentic I-Pdu will be the input to the Authenticator generation algorithm.
SecOCTxSecuredPduLayer	1	This container specifies the Pdu that is transmitted by the Sec OC module to the PduR after the Mac was generated.

⌋

For parameter table [ECUC_SecOC_00079] SecOCAuthenticationBuildAttempts, see definition below container SecOCRxPduProcessing.

For parameter table [ECUC_SecOC_00095] SecOCAuthInfoTruncLength, see definition below container SecOCRxPduProcessing.

For parameter table [ECUC_SecOC_00014] SecOCDataId, see definition below container SecOCRxPduProcessing.

For parameter table [ECUC_SecOC_00021] SecOCFreshnessValueId, see definition below container SecOCRxPduProcessing.

For parameter table [ECUC_SecOC_00015] SecOCFreshnessValueLength, see definition below container SecOCRxPduProcessing.

For parameter table [ECUC_SecOC_00094] SecOCFreshnessValueTruncLength, see definition below container SecOCRxPduProcessing.

[ECUC_SecOC_00084] Definition of EcucBooleanParamDef SecOCProvideTx TruncatedFreshnessValue ⌈

Parameter Name	SecOCProvideTxTruncatedFreshnessValue		
Parent Container	SecOCTxPduProcessing		
Description	This parameter specifies if the Tx query freshness function provides the truncated freshness info instead of generating this by SecOC In this case, SecOC shall add this data to the Authentic PDU instead of truncating the freshness value.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

⌋

[ECUC_SecOC_00103] Definition of EcucBooleanParamDef SecOCReAuthenticateAfterTriggerTransmit ⌈

Parameter Name	SecOCReAuthenticateAfterTriggerTransmit		
Parent Container	SecOCTxPduProcessing		
Description	This parameter specifies if the authentication information of the Secured PDU is updated after the successful transmission of a triggered transmission was confirmed. TRUE if the authentication information shall be updated after triggered transmission. FALSE if the authentication information shall not be updated after triggered transmission. Note: This parameter should only be set to FALSE if the upper layer SecOC_IfTransmit have the same or a higher frequency than the SecOC_TriggerTransmit calls.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	true		
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

⌋

[ECUC_SecOC_00101] Definition of EcucIntegerParamDef SecOCTxPduUnusedAreasDefault ⌈

Parameter Name	SecOCTxPduUnusedAreasDefault		
Parent Container	SecOCTxPduProcessing		
Description	The AUTOSAR SecOC module fills not used areas of a transmitted Secured Pdu or a transmitted Cryptographic Pdu with this byte pattern. This attribute is mandatory to avoid undefined behavior.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 255		
Default value	0		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	–	
Scope / Dependency	scope: local		

⌋

[ECUC_SecOC_00085] Definition of EcucBooleanParamDef SecOCUseTxConfirmation ⌈

Parameter Name	SecOCUseTxConfirmation		
Parent Container	SecOCTxPduProcessing		
Description	A Boolean value that indicates if the function SecOC_SPduTxConfirmation shall be called for this PDU.		
Multiplicity	0..1		
Type	EcucBooleanParamDef		
Default value	false		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

⌋

For parameter table [ECUC_SecOC_00010] SecOCSameBufferPduRef, see definition below container SecOCRxPduProcessing.

[ECUC_SecOC_00013] Definition of EcucReferenceDef SecOCTxAuthServiceConfigRef ⌈

Parameter Name	SecOCTxAuthServiceConfigRef
Parent Container	SecOCTxPduProcessing
Description	This reference is used to define which crypto service function is called for authentication. If PDUs with a dynamic length are used (e.g. CanTP or Dynamic Length PDUs) a MAC algorithm has to be chosen, that is not vulnerable to length extension attack (e.g. CMAC/HMAC).
Multiplicity	1
Type	Symbolic name reference to CsmJob
Post-Build Variant Value	false
Scope / Dependency	

⌋

[ECUC_SecOC_00111] Definition of EcucReferenceDef SecOCTxPduMainFunctionRef ⌈

Parameter Name	SecOCTxPduMainFunctionRef		
Parent Container	SecOCTxPduProcessing		
Description	Reference to the SecOC_MainFunctionTx this PDU belongs to. Mandatory, if multiple main functions are defined.		
Multiplicity	0..1		
Type	Reference to SecOCMainFunctionTx		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

⌋

10.2.15 SecOCTxAuthenticPduLayer

[ECUC_SecOC_00023] Definition of EcucParamConfContainerDef SecOCTxAuthenticPduLayer ⌈

Container Name	SecOCTxAuthenticPduLayer
Parent Container	SecOCTxPduProcessing
Description	This container specifies the Pdu that is received by the SecOC module from the PduR. For this Pdu the Mac generation is provided.
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
SecOCPduType	1	[ECUC_SecOC_00075]
SecOCTxAuthenticLayerPduId	1	[ECUC_SecOC_00026]
SecOCTxAuthenticLayerPduRef	1	[ECUC_SecOC_00025]

No Included Containers

⌋

For parameter table [ECUC_SecOC_00075] SecOCPduType, see definition below container SecOCRxAuthenticPduLayer.

[ECUC_SecOC_00026] Definition of EcucIntegerParamDef SecOCTxAuthenticLayerPduId ⌈

Parameter Name	SecOCTxAuthenticLayerPduId		
Parent Container	SecOCTxAuthenticPduLayer		
Description	PDU identifier assigned by SecOC module. Used by PduR for SecOC_[If|Tp]Transmit.		
Multiplicity	1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 65535		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local withAuto = true		

⌋

[ECUC_SecOC_00025] Definition of EcucReferenceDef SecOCTxAuthenticLayerPduRef ⌈

Parameter Name	SecOCTxAuthenticLayerPduRef		
Parent Container	SecOCTxAuthenticPduLayer		
Description	Reference to the global Pdu.		
Multiplicity	1		
Type	Reference to Pdu		
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

⌋

10.2.16 SecOCTxSecuredPduLayer

[ECUC_SecOC_00024] Definition of EcucChoiceContainerDef SecOCTxSecuredPduLayer ⌈

Choice Container Name	SecOCTxSecuredPduLayer
Parent Container	SecOCTxPduProcessing
Description	This container specifies the Pdu that is transmitted by the SecOC module to the PduR after the Mac was generated.

No Included Parameters

Container Choices		
Container Name	**Multiplicity**	**Scope / Dependency**
SecOCTxSecuredPdu	0..1	This container specifies one Pdu that is transmitted by the Sec OC module to the PduR after the Mac was generated. This Pdu contains the cryptographic information.
SecOCTxSecuredPduCollection	0..1	This container specifies the Pdu that is transmitted by the Sec OC module to the PduR after the Mac was generated. Two separate Pdus are transmitted to the PduR: Authentic I-PDU and Cryptographic I-PDU.

⌋

10.2.17 SecOCTxSecuredPdu

[ECUC_SecOC_00070] Definition of EcucParamConfContainerDef SecOCTxSecuredPdu ⌈

Container Name	SecOCTxSecuredPdu
Parent Container	SecOCTxSecuredPduLayer
Description	This container specifies one Pdu that is transmitted by the SecOC module to the PduR after the Mac was generated. This Pdu contains the cryptographic information.
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
SecOCAuthPduHeaderLength	0..1	[ECUC_SecOC_00093]
SecOCTxSecuredLayerPduId	1	[ECUC_SecOC_00028]
SecOCTxSecuredLayerPduRef	1	[ECUC_SecOC_00027]

No Included Containers

⌋

For parameter table [ECUC_SecOC_00093] SecOCAuthPduHeaderLength, see definition below container SecOCRxAuthenticPdu.

[ECUC_SecOC_00028] Definition of EcucIntegerParamDef SecOCTxSecuredLayerPduId ⌈

Parameter Name	SecOCTxSecuredLayerPduId		
Parent Container	SecOCTxSecuredPdu		
Description	PDU identifier assigned by SecOC module. Used by PduR for confirmation (SecOC_[If|Tp]TxConfirmation) and for TriggerTransmit.		
Multiplicity	1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 65535		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local withAuto = true		

⌋

[ECUC_SecOC_00027] Definition of EcucReferenceDef SecOCTxSecuredLayerPduRef ⌈

Parameter Name	SecOCTxSecuredLayerPduRef		
Parent Container	SecOCTxSecuredPdu		
Description	Reference to the global Pdu.		
Multiplicity	1		
Type	Reference to Pdu		
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

⌋

10.2.18 SecOCTxSecuredPduCollection

[ECUC_SecOC_00071] Definition of EcucParamConfContainerDef SecOCTxSecuredPduCollection ⌈

Container Name	SecOCTxSecuredPduCollection
Parent Container	SecOCTxSecuredPduLayer
Description	This container specifies the Pdu that is transmitted by the SecOC module to the PduR after the Mac was generated. Two separate Pdus are transmitted to the PduR: Authentic I-PDU and Cryptographic I-PDU.
Configuration Parameters	

No Included Parameters

Included Containers		
Container Name	Multiplicity	Scope / Dependency
SecOCTxAuthenticPdu	1	This container specifies the PDU (that is transmitted by the Sec OC module to the PduR) which contains the Secured I-PDU Header and the Authentic I-PDU.
SecOCTxCryptographicPdu	1	This container specifies the Cryptographic Pdu that is transmitted by the SecOC module to the PduR after the Mac was generated.
SecOCUseMessageLink	0..1	SecOC links an Authentic I-PDU and Cryptographic I-PDU together by repeating a specific part (Message Linker) of the Authentic I-PDU in the Cryptographic I-PDU.

⌋

10.2.19 SecOCTxAuthenticPdu

[ECUC_SecOC_00072] Definition of EcucParamConfContainerDef SecOCTxAuthenticPdu ⌈

Container Name	SecOCTxAuthenticPdu
Parent Container	SecOCTxSecuredPduCollection
Description	This container specifies the PDU (that is transmitted by the SecOC module to the Pdu R) which contains the Secured I-PDU Header and the Authentic I-PDU.
Configuration Parameters	

Included Parameters		
Parameter Name	Multiplicity	ECUC ID
SecOCAuthPduHeaderLength	0..1	[ECUC_SecOC_00093]
SecOCTxAuthenticPduId	1	[ECUC_SecOC_00055]
SecOCTxAuthenticPduRef	1	[ECUC_SecOC_00056]

No Included Containers

⌋

For parameter table [ECUC_SecOC_00093] SecOCAuthPduHeaderLength, see definition below container SecOCRxAuthenticPdu.

[ECUC_SecOC_00055] Definition of EcucIntegerParamDef SecOCTxAuthenticPduId ⌈

Parameter Name	SecOCTxAuthenticPduId		
Parent Container	SecOCTxAuthenticPdu		
Description	PDU identifier of the Authentic I-PDU assigned by SecOC module. Used by PduR for confirmation (SecOC_IfTxConfirmation) and for TriggerTransmit.		
Multiplicity	1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 65535		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local withAuto = true		

⌋

[ECUC_SecOC_00056] Definition of EcucReferenceDef SecOCTxAuthenticPduRef ⌈

Parameter Name	SecOCTxAuthenticPduRef		
Parent Container	SecOCTxAuthenticPdu		
Description	Reference to the global Pdu.		
Multiplicity	1		
Type	Reference to Pdu		
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

⌋

10.2.20 SecOCTxCryptographicPdu

[ECUC_SecOC_00073] Definition of EcucParamConfContainerDef SecOCTxCryptographicPdu ⌈

Container Name	SecOCTxCryptographicPdu
Parent Container	SecOCTxSecuredPduCollection
Description	This container specifies the Cryptographic Pdu that is transmitted by the SecOC module to the PduR after the Mac was generated.
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
SecOCTxCryptographicPduId	1	[ECUC_SecOC_00057]
SecOCTxCryptographicPduRef	1	[ECUC_SecOC_00058]

No Included Containers

⌋

[ECUC_SecOC_00057] Definition of EcucIntegerParamDef SecOCTxCryptographicPduId ⌈

Parameter Name	SecOCTxCryptographicPduId		
Parent Container	SecOCTxCryptographicPdu		
Description	PDU identifier of the Cryptographic I-PDU assigned by SecOC module. Used by PduR for confirmation (SecOC_IfTxConfirmation) and for TriggerTransmit.		
Multiplicity	1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 65535		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local withAuto = true		

⌋

[ECUC_SecOC_00058] Definition of EcucReferenceDef SecOCTxCryptographicPduRef ⌈

Parameter Name	SecOCTxCryptographicPduRef		
Parent Container	SecOCTxCryptographicPdu		
Description	Reference to the global Pdu.		
Multiplicity	1		
Type	Reference to Pdu		
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

⌋

10.2.21 SecOCUseMessageLink

[ECUC_SecOC_00074] Definition of EcucParamConfContainerDef SecOCUseMessageLink ⌈

Container Name	SecOCUseMessageLink
Parent Container	SecOCRxSecuredPduCollection, SecOCTxSecuredPduCollection
Description	SecOC links an Authentic I-PDU and Cryptographic I-PDU together by repeating a specific part (Message Linker) of the Authentic I-PDU in the Cryptographic I-PDU.
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
SecOCMessageLinkLen	1	[ECUC_SecOC_00060]
SecOCMessageLinkPos	1	[ECUC_SecOC_00059]

No Included Containers

⌋

[ECUC_SecOC_00060] Definition of EcucIntegerParamDef SecOCMessageLinkLen ⌈

Parameter Name	SecOCMessageLinkLen		
Parent Container	SecOCUseMessageLink		
Description	Length of the Message Linker inside the Authentic I-PDU in bits.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 65535		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

⌋

[ECUC_SecOC_00059] Definition of EcucIntegerParamDef SecOCMessageLinkPos ⌈

Parameter Name	SecOCMessageLinkPos		
Parent Container	SecOCUseMessageLink		
Description	The position of the Message Linker inside the Authentic I-PDU in bits. The bit counting is done according to 01068 and the bit ordering is done according to TPS_SYST_01069.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 65535		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

⌋

10.2.22 SecOCTxPduSecuredArea

[ECUC_SecOC_00086] Definition of EcucParamConfContainerDef SecOCTxPduSecuredArea ⌈

Container Name	SecOCTxPduSecuredArea
Parent Container	SecOCTxPduProcessing
Description	This container specifies an area in the Authentic I-Pdu that will be the input to the Authenticator generation algorithm. If this container does not exist in the configuration the complete Authentic I-Pdu will be the input to the Authenticator generation algorithm.
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
SecOCSecuredTxPduLength	1	[ECUC_SecOC_00088]
SecOCSecuredTxPduOffset	1	[ECUC_SecOC_00087]

No Included Containers

⌋

[ECUC_SecOC_00088] Definition of EcucIntegerParamDef SecOCSecuredTxPduLength ⌈

Parameter Name	SecOCSecuredTxPduLength		
Parent Container	SecOCTxPduSecuredArea		
Description	This parameter defines the length (in bytes) of the area within the Pdu which shall be secured		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	1 .. 4294967295		
Default value	–		
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

⌋

[ECUC_SecOC_00087] Definition of EcucIntegerParamDef SecOCSecuredTxPduOffset ⌈

Parameter Name	SecOCSecuredTxPduOffset		
Parent Container	SecOCTxPduSecuredArea		
Description	This parameter defines the start position (offset in bytes) of the area within the Pdu which shall be secured		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 4294967295		
Default value	0		
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

⌋

10.2.23 SecOCRxPduSecuredArea

[ECUC_SecOC_00089] Definition of EcucParamConfContainerDef SecOCRxPduSecuredArea ⌈

Container Name	SecOCRxPduSecuredArea
Parent Container	SecOCRxPduProcessing
Description	This container specifies an area in the Authentic I-Pdu that will be the input to the Authenticator verification algorithm. If this container does not exist in the configuration the complete Authentic I-Pdu will be the input to the Authenticator verification algorithm.
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
SecOCSecuredRxPduLength	1	[ECUC_SecOC_00091]
SecOCSecuredRxPduOffset	1	[ECUC_SecOC_00090]

No Included Containers

⌋

[ECUC_SecOC_00091] Definition of EcucIntegerParamDef SecOCSecuredRxPdu Length ⌈

Parameter Name	SecOCSecuredRxPduLength		
Parent Container	SecOCRxPduSecuredArea		
Description	This parameter defines the length (in bytes) of the area within the Pdu which is secured		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	1 .. 4294967295		
Default value	–		
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

⌋

[ECUC_SecOC_00090] Definition of EcucIntegerParamDef SecOCSecuredRxPdu Offset ⌈

Parameter Name	SecOCSecuredRxPduOffset		
Parent Container	SecOCRxPduSecuredArea		
Description	This parameter defines the start position (offset in bytes) of the area within the Pdu which is secured		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 4294967295		
Default value	0		
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE

▽

△

	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

⌋

10.3 Published Information

For details refer to the chapter 10.3 “Published Information” in [4].

A Annex A: Application hints for the development of SW-C Freshness Value Manager

A.1 Overview of freshness value construction

The freshness value is provided to SecOC either by a SW-C or CD. SecOC specification provides the required interfaces to request the freshness value either for transmission or for reception of a Secured I-PDU and the required interfaces to propagate the information of a failed or successful transmission or reception. There are several ways to construct and synchronize freshness value across ECUs.

This chapter specifies four use cases ([UC_SecOC_00200], [UC_SecOC_00201], [UC_SecOC_00202], [UC_SecOC_00203]) that describe different ways how a freshness value shall be constructed.

A.2 Freshness Value Based on Single Freshness Counter

[UC_SecOC_00200] ⌈The Software Component Freshness Value Manager (FVM) shall provide the Freshness Value (FV) to SecOC.

The FV construction is based on Freshness Counters realized by means of individual message counters.⌋

The FVM shall provide a Freshness Counter for each configured Freshness Value ID (parameter `SecOCFreshnessValueId`.

Construction

When using a Freshness Counter instead of a Timestamp, the Freshness Counter is incremented prior to providing the authentication information to SecOC on the receiver side.

To properly ensure freshness, the Freshness Counter on both sides of the communication channel should be incremented synchronically.

The Freshness Counter has to be incremented for each outgoing message that is intended to be recognized as an individual incoming message on the receiver side. On the receiver side, the MAC verification of each received message including the counter update shall be performed exactly once.

The FVM shall increment the Freshness Counter corresponding to `SecOCFreshnessValueId` by 1 (CNT ++) only if SecOC has started the transmission of the Secured I-PDU by calling the PduR for further routing.

If the transmission of the Secured I-PDU has been cancelled before, FVM should not increment the Freshness Counter corresponding to SecOCFreshnessValueId.

Please note that when Freshness Counters are used as a FV, the FVM may allow the usage of second Freshness Values.

Verification of I-PDUs

The FVM module shall construct Freshness Verify Value (i.e. the Freshness Value to be used for Verification) and provide it to SecOC. In the event the complete Freshness Value is transmitted in the secured I-PDU, it needs to be verified that the constructed FreshnessVerifyValue is larger than the last stored notion of the Freshness Value. If it is not larger than the last stored notion of the Freshness Value, the FVM shall stop the verification and drop the Secured I-PDU.

Otherwise, constructing the Authentication Verify Counter is defined as outlined by the following pseudo code.

```
If (SecOCFreshnessValueTruncLength = FreshnessValueLength)
{
  FreshnessVerifyValue = FreshnessValue parsed from Secured I-PDU;
}
Else
{
  If (FreshnessValue parsed from Secured I-PDU > least significant bits of
      FreshnessValue corresponding to SecOCFreshnessValueId)
  {
    Attempts = 0;
    FreshnessVerifyValue = most significant bits of FreshnessValue
    corresponding to SecOCFreshnessValueId | FreshnessValue parsed
    from Secured I-PDU;
  }
  Else
  {
    Attempts = 0;
    FreshnessVerifyValue = most significant bits of FreshnessValue
    corresponding to SecOCFreshnessValueId + 1 | FreshnessValue parsed
    from payload;
  }
}
```

A.3 Freshness Value Based on Single Freshness Timestamp

[UC_SecOC_00201] ⌈The Software Component Freshness Value Manager (FVM) shall provide the Freshness Value (FV) to SecOC.

The FV construction is based on Freshness Counters realized by means of Timestamps.⌋

Source of global time values

The global synchronized time can be used as base for the Freshness Timestamp,. This global synchronized time will have the same value at the sender and all receivers. Therefore its value can be used as Freshness Value with the advantage that it does not necessarily need to be transmitted within the Secured PDU itself and it does not need to be transmitted for every sender and receiver individually.

Resolution and precision of global time values

The FVM has to consider the resolution and precision of the used global time values.

Please note that when Freshness Timestamps are used as a FV, the FVM may allow the usage of an Acceptance Window mechanism.

Verification of I-PDUs

The SecOC module shall construct Freshness Verify Value (i.e. the Freshness Value to be used for Verification) and provide it to SecOC. In case of complete Freshness Value transmission, it needs to be verified that the constructed FreshnessVerifyValue is within the acceptance window defined by SecOCRxAcceptanceWindow. If it is not in that window, the SecOC module shall stop the verification and drop the Secured I-PDU.

Otherwise, constructing the Authentication Verify Value is defined as outlined by the following pseudo code.

```
If (SecOCFreshnessValueTruncLength = FreshnessValueLength)
{
  FreshnessVerifyValue = FreshnessValue parsed from Secured I-PDU;
}
Else
{
  If ((most significant bits of FreshnessValue corresponding to
       SecOCFreshnessValueId | FreshnessValue parsed from Secured I-PDU) <
       (max(0: (most significant bits of FreshnessValue corresponding to
       SecOCFreshnessValueId | least significant bits of FreshnessValue
       corresponding to SecOCFreshnessValueId) - SecOCRxAcceptanceWindow)))
  {
    Attempts = 0;
    FreshnessVerifyBaseValue = most significant bits of FreshnessValue
    corresponding to SecOCFreshnessValueId + 1;
  }
  Else
  {
    Attempts = 0;
    FreshnessVerifyBaseValue = most significant bits of FreshnessValue
    corresponding to SecOCFreshnessValueId;
  }
```

```
  FreshnessVerifyValue = FreshnessVerifyUpperValue =
  FreshnessVerifyLowerValue = FreshnessVerifyBaseValue | FreshnessValue
  parsed from Secured I-PDU;
}
```

A.4 Freshness Value Based on Multiple Freshness Counters (Prerequisite: Truncated Freshness Value)

[UC_SecOC_00202] ⌈Construction of Freshness value from decoupled counters.

The Freshness Value Manager (FVM) (SW-C or CDD) provide the Freshness Value (FV) to SecOC. FVM supports a master-slave synchronization mechanism for FV in the precondition of truncated freshness value.⌋

The figure below shows the relationship between FV management master ECU and slave (Sender / Receiver) ECU.

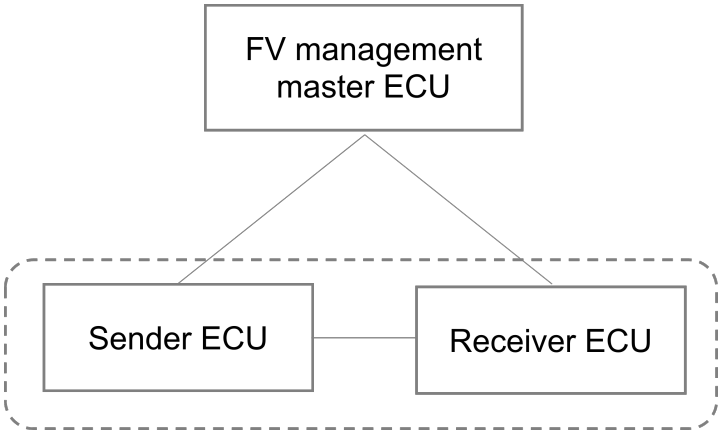

Figure A.1: FvMaster Relationship Sender/Receiver ECU

Entity	*Description*
Sender ECU (Sender)	Sends a Secured I-PDU to the receiver ECU. Receives the synchronization message (TripResetSyncMsg) from the FV management master ECU and constructs the freshness value required to send the Secured I-PDU.
Receiver ECU (Receiver)	Receives a Secured I-PDU. Receives the synchronization message (TripResetSyncMsg) from the FV management master ECU and constructs the freshness value required to verify the received Secured I-PDU.
FV management master ECU (FvMaster)	Sends the synchronization message (TripResetSyncMsg) to all of the sender and receiver ECUs.

Table A.1: FvMaster Relationship Sender/Receiver ECU

FVM shall have a master synchronization function and a slave-transmission synchronization function. This will make it possible to implement the following two FV management master methods.

1. Single FV management master method

In this configuration, the system has only one FV management master ECU. For the system configuration and the entity list, see Figure A.1 and Table A.1, respectively.

2. Multi FV management master method

In this configuration, the system has multiple FV management master ECUs for the same number of sender ECUs. It means that a Sender ECU doubles as the FV management master entity ECU for secured I-PDUs which the Sender ECU manages. The system configuration and the entity list of the multi FV management master method are as follows.

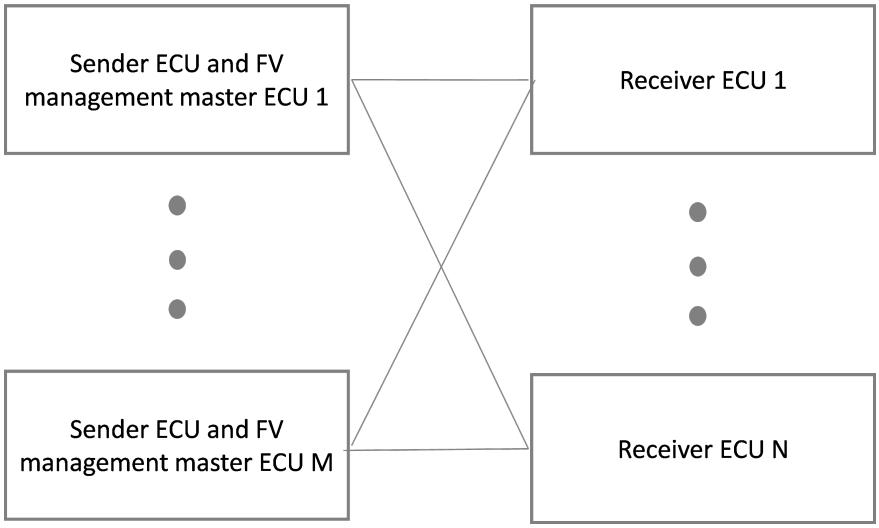

Figure A.2: System Configuration for Multi FV Manager Master Method

Entity	*Description*
Sender ECU and FV management master ECU (Sender&FvMaster)	Sends a Secured I-PDU to the receiver ECU. Sends the synchronization message (TripResetSyncMsg) to the receiver ECU.
Receiver ECU (Receiver)	Receives a Secured I-PDU. Receives the synchronization message(TripResetSyncMsg).

Table A.2: Entity List for Multi FV Manager Master Method

Note:

A receiver ECU receives a synchronization message from a Sender ECU which sends secured I-PDUs which the receiver wants to get. If it receives messages from multiple sender ECUs, then it receives synchronization messages from the multiple sender ECUs.

A.4.1 Definition of Freshness Value

A.4.1.1 Structure of Freshness Value

Software Component FVM provides the FV to SecOC constructed from separate counters in the following structure:

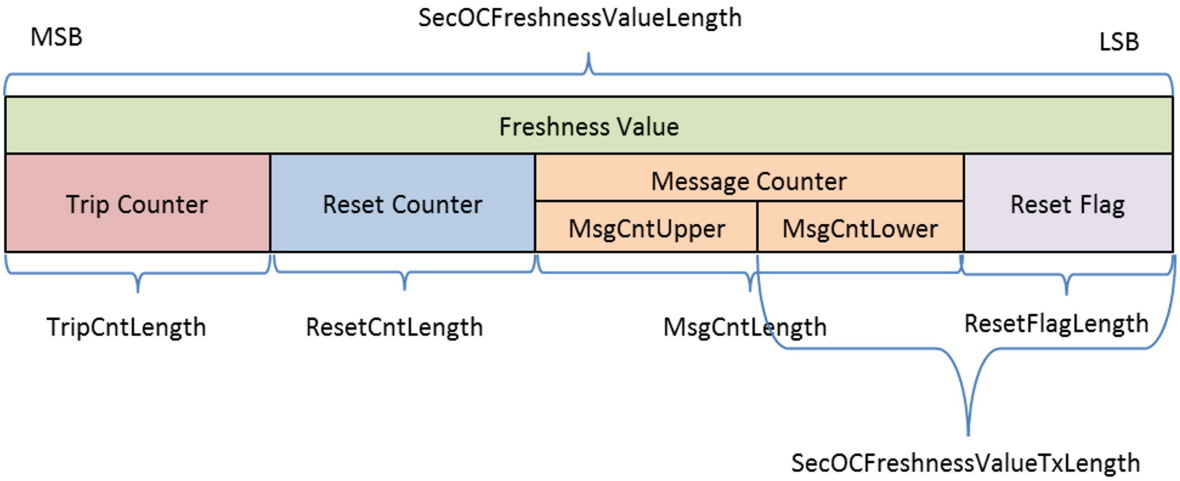

Figure A.3: Structure of FreshnessValue

Data	***Description***
Trip Counter (TripCnt)	This counter is incremented in units of trips by the FV management master ECU. With the single FV management master method, the FV management master ECU sends a new TripCnt as the synchronization message (TripResetSyncMsg) to the sender ECU and receiver ECU. All the sender and receiver ECUs maintain this value. With the multi FV management master method, the sender ECU sends a new TripCnt as the synchronization message to the receiver ECU. The receiver ECU maintains this value.
Reset counter (ResetCnt)	This counter is incremented periodically by the FV management master ECU on the cycle configured by ResetCycle. With the single FV management master method, the FV management master ECU sends a new ResetCnt as the synchronization message (TripResetSyncMsg) to the sender ECU and receiver ECU. All of the sender and receiver ECUs maintain this value. With the multi FV management master method, the sender ECU sends a new ResetCnt as the synchronization message to the receiver ECU. The receiver ECU maintains this value.
Message counter (MsgCnt)	This counter is incremented with every message transmission by the sender ECU. It is managed for each secure message by the sender ECU. "MsgCntLower" refers to the range that is included in the truncated freshness value for Message Counter transmission (inside SecOCFreshnessValueTxLength). "MsgCntUpper" refers to the range that is not included in the truncated freshness value for Message Counter transmission (outside SecOCFreshnessValueTxLength).
Reset Flag (ResetFlag)	This flag is updated in synchronization with the reset counter. It is the ResetFlagLength(bit) value from the lower end of the reset counter.

Table A.3: Structure of Freshness Value

Abbreviation	*Description*
ResetCycle	Reset counter increment cycle
TripCntLength	Full length of the trip counter (bit)
ResetCntLength	Full length of the reset counter (bit)
MsgCntLength	Full length of the message counter (bit)
ResetFlagLength	Length of the reset flag (bit)
ClearAcceptanceWindow	Permissible range for a counter initialization when the trip counter reaches the maximum value. Under the erroneous situation such as miss-synchronous counter between FV master and slave around upper limit of trip counter, this window parameter would work effectively to recover the situation as a robustness. To understand further mechanism, see clause A.4.1.2.

Table A.4: Abbreviation of FVM variable

A.4.1.2 Specification of counters used to construct Freshness Value

Counter	*Increment condition*	*Initialization condition*	*Initial value*	*Counter length*
Trip counter (TripCnt)	- When the FV management master ECU starts - On wakeup - On reset - When the power status changes: "IG-OFF=>IG-ON", incremented by 1	The increment conditions occur at the maximum value of the trip counter.	FV management master ECU: 1 Slave ECU: 0	TripCntLength Max 24 bit
Reset counter (ResetCnt)	Incremented by 1 at regular time intervals (ResetCycle)	The trip counter is incremented or initialized.	FV management master ECU: 1 Slave ECU: 0	ResetCntLength Max 24bit
Message counter (MsgCnt)	Increment 1 value for each message transmission	The reset counter is incremented or initialized.	Slave ECU: 0	MsgCntLength Max 48 bit
Reset Flag (ResetFlag)	- (It follows the reset counter, as it is the ResetFlagLength(bit) value from the lower end of the reset counter.)			ResetFlagLength Max 2bit

Table A.5: Behavior of counters used to construct freshness value

Note: The Length of Freshness Value (SecOCFreshnessValueLength) cannot exceed 64 bits, so the lengths of each of the three counters (Trip Counter, Reset Counter, Message Counter) and reset flag must be adopted individually, to match this requirement that their total length does not exceed 64 bits.

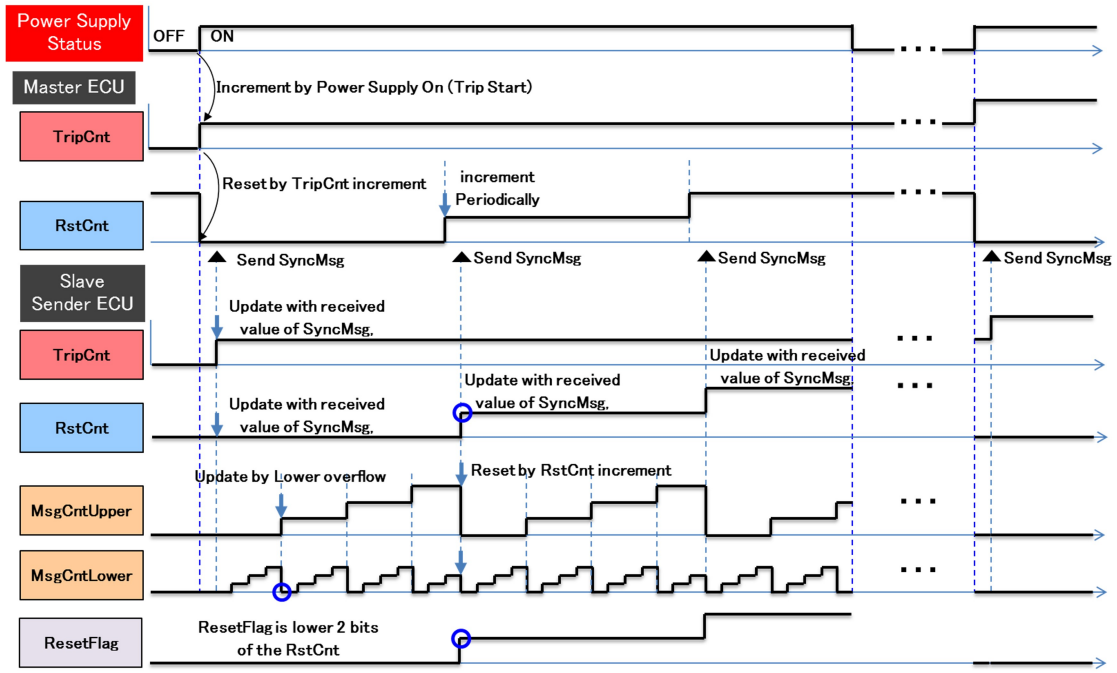

Figure A.4: Behavior example of freshness value (TripCnt, RstCnt, MsgCnt, ResetFlag)

Note:

Figure A.4 shows an example of the case where "ResetFlagLength is 2 bits, and MsgCntLower is 2 bits". Be careful to design the counter values whose maximum is never reached in order to prevent attacks such as replay.

If each of the counters that constitute the freshness value reaches its maximum value, the following procedures are taken. In addition, the slave ECU notifies the upstream module that the message counter value has reached the maximum value.

[Reason] Even when one of the counters that constitute the freshness value reaches its maximum value, it may still be desirable to continue the communication.

[Reason] When any counter reaches its maximum value, replay attacks can no longer be detected.

1. FV management master ECU

- At the maximum value of the trip counter

  When an increment condition of the trip counter occurs at the maximum value of the trip counter, the trip counter and the reset counter are returned to their initial values. The synchronization message is sent even after the trip counter is returned to the initial value.

- At the maximum value of the reset counter

  When an increment condition of the reset counter occurs at the maximum value of the reset counter, the reset counter is fixed to the maximum value.

The synchronization message is sent even at the maximum value of the reset counter. Even though FV is still overflowed notifying to upper layer application or diagnostic system, there are some use case which wants to continue to communicate with other ECUs under limited circumstance. For the purpose of synchronization with the Slave ECU, FV Master is fixing the counter value on the upper limit to wait for re-sync from Slave ECU side, thus master ECU periodically try to send TripResetSyncMsg with fixed RstCnt until re-sync succeeds.

2. Slave ECU

- At the maximum value of the trip counter

  If both Conditions 1 and 2 below are established, the synchronization message is received and authenticator verification is performed.

  If the verification result is OK, the latest values of the trip counter and reset counter are updated with the received trip counter and reset counter values.

  In addition, the previously sent value and previously received value of each counter are returned to the initial values.

  Condition 1:

  "Maximum value of the trip counter" - "ClearAcceptanceWindow"

  $\leq$ "Latest value of the trip counter maintained by the slave ECU"

  $\leq$ "Maximum value of the trip counter"

  Condition 2:

  "Initial value of the trip counter"

  $\leq$ "Trip counter value in the synchronization message"

  $\leq$ "Initial value of the trip counter" + "ClearAcceptanceWindow"

  [Reason] This is to provide a permissible range (ClearAcceptanceWindow), taking into consideration cases where the trip counters of the FV management master ECU and slave ECU deviate from each other around the maximum value. The initial value of the trip counter in Condition 2 refers to the initial value of the FV management master ECU.

- At the maximum value of the reset counter

  The sender ECU generates an authenticator by fixing the message counter to the maximum value.

  The receiver ECU verifies the authenticator by overwriting the message counter with the maximum value.

- At the maximum value of the message counter

  The sender ECU generates an authenticator by fixing the message counter to the maximum value.

The receiver ECU verifies the authenticator by overwriting the message counter with the maximum value.

A.4.2 Synchronization Message Format

The FV management master ECU and slave ECU handle the synchronization messages that comply with the following format.

Note:

The message used to synchronize the trip counter with the reset counter is sent from the FV management master ECU to the slave ECU. It is desirable to use the same message for the trip counter and reset counter.

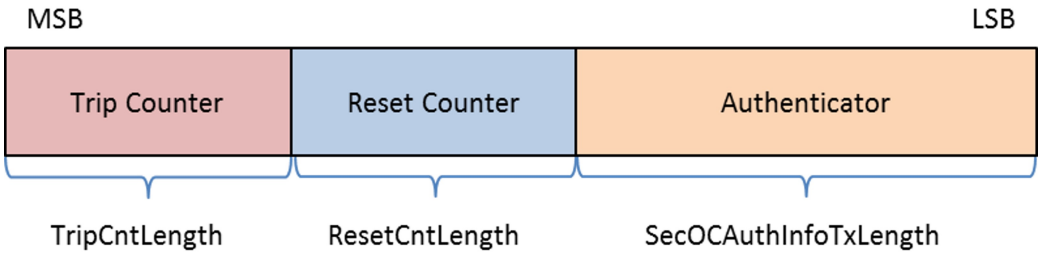

Figure A.5: Format of the synchronization message (TripResetSyncMsg)

A.4.3 Processing of FV Management Master

A.4.3.1 Processing of Initialization

The FV management master ECU performs the following processes at ECU startup, on wakeup or ECU reset.

- Obtain the trip counter value that is stored in the nonvolatile memory.

  Set the trip counter to the initial value at the first startup.

- Set the reset counter to the initial value.

When the trip counter value cannot be read from the non-volatile memory, any failsafe value can be used as the trip counter and reset counter until the next trip counter update.

When the trip counter is incremented, the FV management master ECU stores the incremented value to the nonvolatile memory. It might be better that the trip counter is stored in secure flash to prevent from malicious manipulation as an option, using RAM buffering. However, storing the failsafe value into the non-volatile memory shall not be implemented.

Note:

Even when the trip counter changes from the maximum value to the initial value (see clause A.4.1.2), it is treated as an increment and is stored in the non-volatile memory.

A.4.3.2 Sending of Synchronization Message

The FV management master ECU sends the trip counter and reset counter that it manages to the slaveECU periodically (every ResetCycle). However, if they can be sent at startup, it sends them immediately.

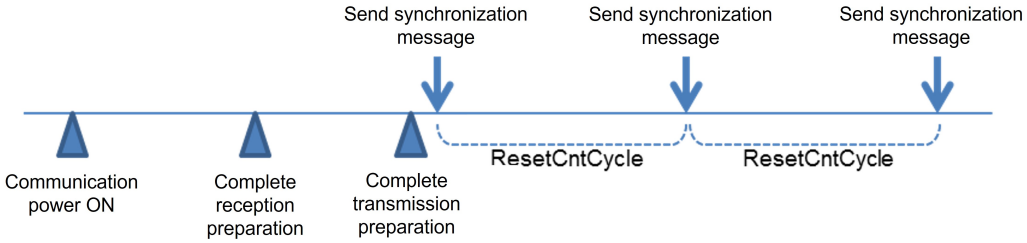

Figure A.6: Transmission Timing of Synchronization Message

A.4.4 Processing of Slave ECUs

Software Component Freshness Value Manager [FVM] shall implement and store the following design values for counters:

Design Value	*Description*	*Update condition*
Trip Counter (TripCnt)	Latest Trip Counter value received successfully from FV management master ECU.	Successful reception of TripResetSyncMsg
Reset Counter (ResetCnt)	Latest Reset Counter value received successfully from FV management master ECU	Successful reception of TripResetSyncMsg
Freshness Value (FV)	Freshness Value maintained for each message to be secured. The structure of FV is according to Figure A.3. Transmission message: Before a Secured I-PDU is sent (when SecOC requests FV to be provided), it holds the value used in the transmission of the previous Secured I-PDU. After it is sent (when SecOC sends a notification of the transmission of the Secured I-PDU), the value is updated with the value provided to SecOC for transmission. Reception message: Before a Secured I-PDU is received (when SecOC requests FV to be provided), it holds the value used for verification at the reception of the previous Secured I-PDU. After it is received (when SecOC sends a notification of successful MAC verification), the value is updated with the value provided to SecOC for reception.	SecOC notification of the start of Secured I-PDU transmission or, SecOC notification of successful MAC verification

Table A.6: Design Value for Counter

Explanation:

- Latest Trip Counter or Reset Counter refers to the values received from FV management master ECU
- Previous Trip Counter, Reset Counter, Message Counter and Reset Flag refers to the individual freshness values used for previous authentication generation or verification.
- Received Reset Flag or Message Counter refers to truncated freshness value used to build the Authentication Information as described by SecOC.

Trip Counter and Reset Counter provided by FV management master ECU and stored by FVM.

Trip Counter [TripCnt] **Latest**	Reset Counter [ResetCnt] **Latest**

Table A.7: Trip Counter and Reset Counter

Freshness Value for each secured I-PDU that is provided to SecOC by FVM and it consists of Trip Counter, Reset Counter, Message Counter, Reset Flag.

Trip Counter [TripCntPdu] **Previous**	Reset Counter [ResetCntPdu] **Previous**	Message Counter [MsgCnt] **Previous**	Reset Flag [ResetFlag] **Previous**
Freshness Value [FV] For each secured message. It holds the value used for previous transmission or reception of a secured I-PDU.			

Table A.8: Freshness Value for each secured message

A.4.4.1 Processing of Initialization

The slave ECU performs the following processes at ECU startup, on wakeup or ECU reset.

- Obtain the trip counter value that is stored in the non-volatile memory, and then set it to the latest value.

  Set the initial value to the latest value of the trip counter at the first startup, or when the trip counter value cannot be read from the non-volatile memory.
- Set the latest value of the reset counter to the initial value.
- Set all the previously sent values and previously received values to the initial values.

Note:

The latest value of the trip counter has been saved in the non-volatile memory. Both latest and previous trip value in volatile memory are initialized based on the trip counter

in the non-volatile memory. In this context, the previous value refers to the previously sent value for the sender ECU, or the previously received value for the receiver ECU.

A.4.4.2 Receiving of Synchronization Message

When the synchronization message is received, the slave ECU performs the following processes to complete the synchronization process.

1. SecOC obtains the freshness value for verification from FVM. FVM compares the freshness value in the method described in [UC_SecOC_00200], and constructs the freshness value for verification, because it is assumed that the trip counter value and reset counter value in the synchronization message (TripResetSyncMsg) are sent and received at full length.
2. SecOC constructs the authentication data, which consists of "Message ID | Freshness value".

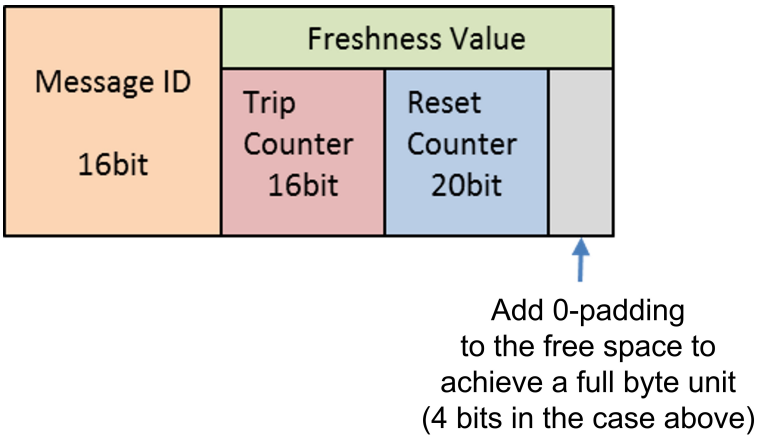

Figure A.7: Example of Authentication Data Structure of Synchronization Message (TripResetSyncMsg)

3. SecOC verifies the authenticator and notifies FVM of the verification result. If the verification result is OK, FVM updates the received trip counter value and reset counter value as the latest values. SecOC also notifies the application of the received trip counter value and reset counter value.
4. If the verification result fails (NG), SecOC does not perform re-verification, but notifies the application and discards the reception message.

Note:

When the trip counter is incremented, the application stores the incremented value to the non-volatile memory. It is preferable that the value is stored securely. However, storing the failsafe value into the non-volatile memory shall not be implemented. Even when the trip counter changes from the maximum value to the initial value (see clause A.4.1.2), it is treated as an increment and is stored in the non-volatile memory.

A.4.4.3 Construction of Freshness Value for Transmission

When SecOC requests to obtain the freshness value for transmission, FVM constructs the freshness value for transmission according to Table A.9.

Trip Counter | Reset Counter comparison (*1)	Construction of Freshness Value for Transmission			
	Trip Counter	Reset Counter	Message Counter	Reset Flag
Latest value= Previously sent value	Previously sent value	Previously sent value	Previously sent value +1	The ResetFlagLength(bit) value from the lower end of the reset counter (previously sent value)
Latest value ≠ Previously sent value	Latest value	Latest value	Inittial Value +1	The ResetFlagLength(bit) value from the lower end of the reset counter (latest value)
*1 - Compare the latest values and previously sent values of the trip counter and reset counter. The "|" symbol means a connection.				

Table A.9: Construction of Freshness Value (FV) for Tx

When SecOC sends a transmission start notification, FVM maintains the constructed freshness value for transmission (trip counter, reset counter, message counter) as the previously sent value.

A.4.4.4 Construction of Freshness Value for Reception

When SecOC requests to obtain the freshness value for verification, FVM constructs the freshness value for verification according to Table A.10, based on the following three results.

1. Reset flag comparison (see Figure A.10)
2. Trip counter and reset counter comparison
3. Message counter (lower end) comparison

Construction Format	Condition			Construction of freshness value for verification			
	(1) Reset flag comparison	(2) Trip counter | reset counter comparison	(3) Message counter (lower end) comparison (*3)	Trip Counter	Reset Counter	Message Counter (Upper) (*1)	Message Counter (Lower) (*2)
Format 1	Latest value = Received value	Latest value = Previously received value	Previously received value < Received value (no carry)	Previously Received value	Previously Received value	Previously Received value	Received value

▽

△

Construction Format	Condition			Construction of freshness value for verification			
	(1) Reset flag comparison	(2) Trip counter | reset counter comparison	(3) Message counter (lower end) comparison (*3)	Trip Counter	Reset Counter	Message Counter (Upper) (*1)	Message Counter (Lower) (*2)
Format 2			Previously received value >= Received value (with carry)	Previously Received value	Previously Received value	Previously received value+1	Received value
Format 3		Latest value > Previously received value	-	Latest value	Latest value	0	Received value
Format 1	Latest value-1 = Received value	Latest value-1 = Previously received value	Previously received value < Received value (no carry)	Previously Received value	Previously Received value	Previously Received value	Received value
Format 2			Previously received value >= Received value (with carry)	Previously Received value	Previously Received value	Previously received value+1	Received value
Format 3		Latest value-1 > Previously received value	-	Latest value	Latest value-1	0	Received value
Format 1	Latest value+1 = Received value	Latest value+1 = Previously received value	Previously received value < Received value (no carry)	Previously Received value	Previously Received value	Previously Received value	Received value
Format 2			Previously received value >= Received value (with carry)	Previously Received value	Previously Received value	Previously received value+1	Received value
Format 3		Latest value+1 > Previously received value	-	Latest value	Latest value+1	0	Received value
Format 1	Latest value-2 = Received value	Latest value-2 = Previously received value	Previously received value < Received value (no carry)	Previously Received value	Previously Received value	Previously Received value	Received value
Format 2			Previously received value >= Received value (with carry)	Previously Received value	Previously Received value	Previously received value+1	Received value
Format 3		Latest value-2 > Previously received value	-	Latest value	Latest value-2	0	Received value
Format 1	Latest value+2 = Received value	Latest value+2 = Previously received value	Previously received value < Received value (no carry)	Previously Received value	Previously Received value	Previously Received value	Received value
Format 2			Previously received value >= Received value (with carry)	Previously Received value	Previously Received value	Previously received value+1	Received value
Format 3		Latest value+2 > Previously received value	-	Latest value	Latest value+2	0	Received value

Note:

(*1) "Message counter (Upper)" refers to the range that is not included in the freshness value of the message counter for transmission.

(*2) "Message counter (Lower)" refers to the range that is included in the freshness value of the message counter for transmission.

(*3) Compare the previously received value of the "message counter (Lower)" with the received value, and determine if carry was produced.

Table A.10: Construction of Freshness Value (FV) for Rx

The sequence for constructing the freshness value for verification, and the reset flag comparison method are as follows.

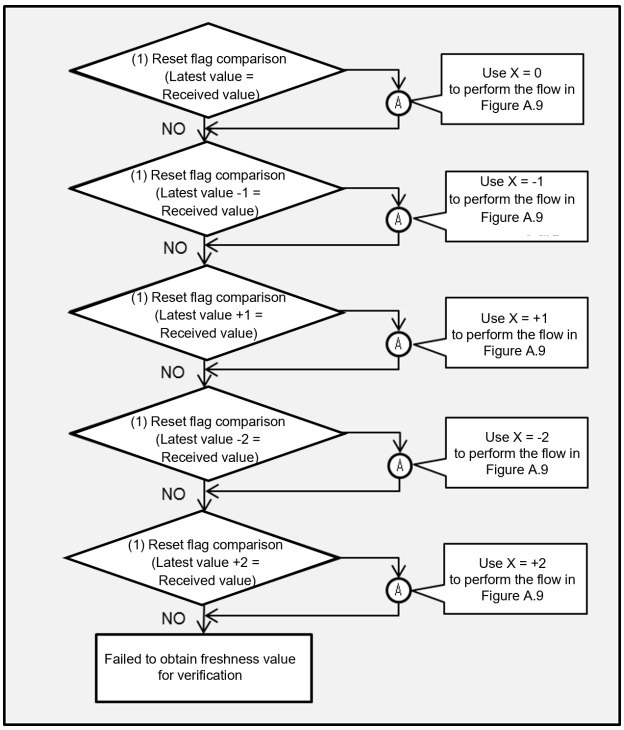

Figure A.8: Construction Order of Freshness Value for Verification 1

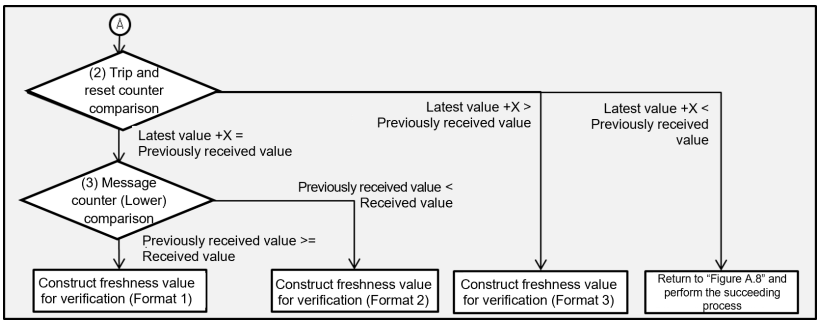

Figure A.9: Construction Order of Freshness Value for Verification 2

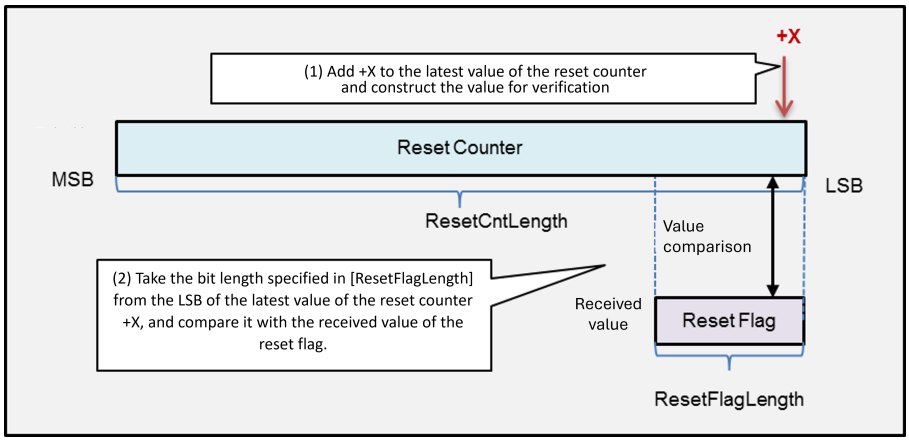

Figure A.10: Reset Flag Comparison Method

SecOC uses the obtained freshness value for verification to construct authentication data and perform authenticator verification. If the verification result fails (NG), SecOC re-constructs the freshness value for verification and performs re-verification.

For the method of constructing the freshness value for verification, see Figure A.8.

When SecOC sends a notification of the verification result (verification = OK), FVM maintains the constructed freshness value for verification as the previously received value.

A.5 Freshness Value Based on Multiple Freshness Counters (Pre-requisite: Complete Freshness Value)

[UC_SecOC_00203] ⌈Construction of Freshness value from decoupled counters.

The Freshness Value Manager (FVM) (SW-C or CDD) provide the Freshness Value (FV) to SecOC. FVM supports a master-slave synchronization mechanism for FV in the precondition of complete freshness value.⌋

The relationship between Sender ECU (and FV management master ECU) and Receiver ECU is same as Figure A.2.

Entity	Description
Sender ECU and FV management master ECU	Sends a Secured I-PDU to the receiver ECU.
Receiver ECU	Receives a Secured I-PDU.

Table A.11: Entity List for Multi FV Manager Master Method

The system has multiple FV management master ECUs for the same number of sender ECUs. It means that a Sender ECU doubles as the FV management master entity ECU for secured I-PDUs which the Sender ECU manages.

Note:

Compared with the Section A.4, the synchronization message is not necessary because the complete freshness value is transmitted and received.

A.5.1 Definition of Freshness Value

A.5.1.1 Structure of Freshness Value

Software Component FVM provides the FV to SecOC constructed from separate counters in the following structure:

Note:

Compared with the Section A.4, the reset counter and the reset flag are not necessary because these are the measure against the gap of freshness value between Sender ECU and Receiver ECU caused by the freshness value truncation.

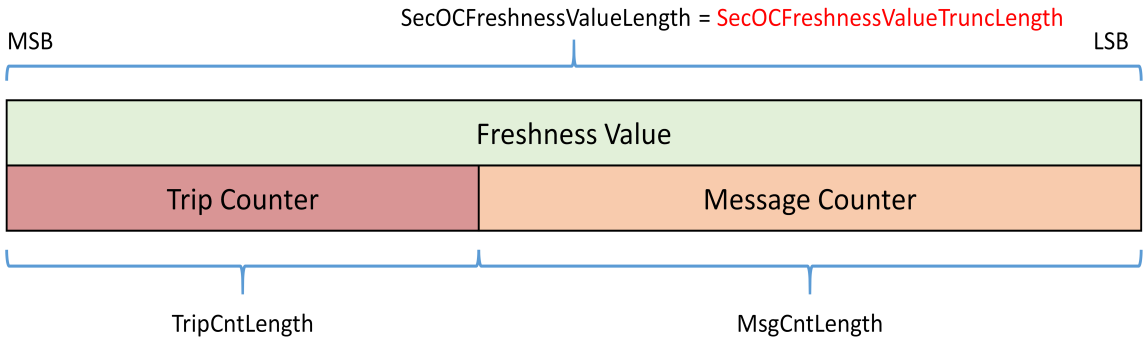

Figure A.11: Structure of FreshnessValue

Data	*Description*
Trip Counter	This counter is incremented in units of trips by sender ECU as the FV management master ECU. It is managed by the sender ECU as the FV management master ECU and set for each sender ECU.
Message counter	This counter is incremented with every message transmission by the sender ECU. It is managed for each secure message by the sender ECU.

Table A.12: Structure of Freshness Value

Abbreviation	*Description*
TripCntLength	Full length of the trip counter (bit)
MsgCntLength	Full length of the message counter (bit)
ClearAcceptanceWindow	Permissible range for a counter initialization when the trip counter reaches the maximum value. Under the erroneous situation such as miss-synchronous counter between sender ECU and receiver ECU around upper limit of trip counter, this window parameter would work effectively to recover the situation as a robustness. To understand further mechanism, see clause A.5.1.2.

Table A.13: Abbreviation of FVM variable

A.5.1.2 Specification of counters used to construct Freshness Value

Counter	*Increment condition*	*Initialization condition*	*Initial value*	*Counter length*
Trip counter	- When the sender ECU starts - On wakeup - On reset - When the power status changes: "IG-OFF=>IG-ON", incremented by 1	The increment conditions occur at the maximum value of the trip counter.	Sender ECU: 1 Receiver ECU: 0	TripCntLength Max 24 bit
Message counter	Increment 1 value for each message transmission	The trip counter is incremented or initialized.	Sender ECU: 0 Receiver ECU: 0	MsgCntLength Max 48 bit

Table A.14: Behavior of counters used to construct freshness value

Note: The Length of Freshness Value (SecOCFreshnessValueLength) cannot exceed 64 bits, so the lengths of each of the two counters (Trip Counter, Message Counter) must be adopted individually, to match this requirement that their total length does not exceed 64 bits.

If each of the counters that constitute the freshness value reaches its maximum value, the following procedures are taken.

[Reason]

- Even when one of the counters that constitute the freshness value reaches its maximum value, it may still be desirable to continue the communication.
- When any counter reaches its maximum value, replay attacks can no longer be detected.

1. Sender ECU

- At the maximum value of the trip counter

  When the trip counter has reached the maximum value, notification is sent to the application that the counter value has reached the maximum. Also when the increment condition for the trip counter is met, the trip counter returns to its initial value.

  A Secured I-PDU is sent even after the trip counter is returned to the initial value.

- At the maximum value of the message counter

  When the message counter has reached the maximum value, notification is sent to the application that the counter value has reached the maximum. Also, the message counter is fixed at the maximum value, and a MAC be generated.

  A Secured I-PDU is sent even after the message counter has reached the maximum.

2. Receiver ECU

- At the maximum value of the trip counter

  When the trip counter has reached the maximum value, notification is sent to the application that the counter value has reached the maximum. Also, if both conditions 1 and 2 below are met, when a Secured I-PDU is received, skip the verification of freshness value (see clause A.5.3.2).

  Condition 1:

  "Maximum value of the trip counter" - "ClearAcceptanceWindow"

  $\leq$ "Trip counter (previously received value) corresponding to the `SecOCFreshnessValueId` that was received and is stored by receiver ECU

  $\leq$ "Maximum value of the trip counter"

  Condition 2:

  "Initial value of the trip counter"

  $\leq$ "Trip counter value in the Secured I-PDU"

  $\leq$ "Initial value of the trip counter" + "ClearAcceptanceWindow"

  [Reason] This is to provide a permissible range (ClearAcceptanceWindow), taking into consideration cases where the trip counters of the sender ECU and receiver ECU deviate from each other around the maximum value. The initial value of the trip counter in Condition 2 refers to the initial value of the sender ECU.

- At the maximum value of the message counter

  When the message counter has reached the maximum value, notification is sent to the application that the counter value has reached the maximum. Also, the message counter is overwritten with the maximum value, and the MAC be verified.

A.5.2 Processing of Sender ECU and FV Management Master ECU

Software Component FVM on the sender ECU and the FV Management Master ECU implements and stores the following design values for counters:

Design Value	*Description*	*Update condition*
Trip Counter (latest value)	Latest Trip Counter value that is incremented each initialization process.	Processing of initialization
Freshness Value (previously sent value)	Freshness Value maintained for each message to be secured. The structure of FV is according to Figure A.11. Transmission message: Before a Secured I-PDU is sent (when SecOC requests FV to be provided), it holds the value used in the transmission of the previous Secured I-PDU. After it is sent (when SecOC sends a notification of the transmission of the Secured I-PDU), the value is updated with the value provided to SecOC for transmission.	SecOC notification of the start of Secured I-PDU transmission

Table A.15: Design Value for Counter

A.5.2.1 Processing of Initialization

The sender ECU performs the following processes at ECU startup, on wakeup or ECU reset.

- The trip counter stored in the non-volatile memory is retrieved and set as the latest value.

  At initial startup, the latest value of the trip counter is set to the initial value.

- Set the initial value to all previously sent values.

When the trip counter value cannot be read from the non-volatile memory, any failsafe value can be used as the trip counter until the next trip counter update.

When the trip counter is incremented, the sender ECU stores the incremented value to the non-volatile memory. It might be better that the trip counter is stored in secure flash to prevent from malicious manipulation as an option, using RAM buffering. However, storing the failsafe value into the non-volatile memory should not be implemented.

Note:

Compared with the Section A.4, the Sender ECU itself manage the trip counter because the Sender ECU doubles as the FV management master ECU.

Even when the trip counter changes from the maximum value to the initial value (see clause A.5.1.2), it is treated as an increment and is stored in the non-volatile memory.

A.5.2.2 Construction of Freshness Value for Transmission

When SecOC requests to obtain the freshness value for transmission, FVM constructs the freshness value for transmission according to Table A.16.

Trip Counter comparison	*Construction of Freshness Value for Transmission*	
	Trip Counter	Message Counter
Latest value = Previously sent value	Previously sent value	Previously sent value +1
Latest value ≠ Previously sent value	Latest value	Inittial Value +1

Table A.16: Construction of Freshness Value (FV) for Tx

When SecOC sends a transmission start notification, FVM maintains the constructed freshness value for transmission (trip counter, message counter) as the previously sent value.

A.5.3 Processing of Receiver ECU

Software Component FVM on the Receiver ECU implements and stores the following design values for counters:

Design Value	*Description*	*Update condition*
Freshness Value (previously received value)	Freshness Value maintained for each message to be secured. The structure of FV is according to Figure A.11. Reception message: Before a Secured I-PDU is received (when SecOC requests FV to be provided), it holds the value used for verification at the reception of the previous Secured I-PDU. After it is received (when SecOC sends a notification of successful MAC verification), the value is updated with the value provided to SecOC for reception.	SecOC notification of successful MAC verification

Table A.17: Design Value for Counter

A.5.3.1 Processing of Initialization

The receiver ECU performs the following processes at ECU startup, on wakeup or ECU reset.

- The trip counter stored in the non-volatile memory is retrieved and set to the value obtained in the corresponding trip counter (previously received value).

  If not storing the trip counter into the non-volatile memory, the trip counter (previously received value) is set to the initial value.

  At initial startup or when the trip counter cannot be read from the non-volatile memory, the trip counter (previously received value) is set to the initial value.

- Set all message counters (previously received value) to the initial value.

- If using the list of previously-received freshness values, the above previously received value is treated as reference value and set tolerance value according to clause A.5.3.3.

Note:

It is assumed that the freshness value is initialized including the non-volatile memory when key updating.

A.5.3.2 Verification of I-PDUs

FVM constructs the freshness value for verification and compares the freshness value in the method described in [UC_SecOC_00200], because the complete freshness value (trip counter and message counter) is transmitted and received.

If there are multiple communication paths in using Ethernet communications, etc, the message frames may arrive out of sequence. The alternative method shown bellows may be used, which is to prevent discarding if out of sequence.

The freshness values that were stored up to this point are included in the list of previously-received freshness values. The reference value refers to the largest value among the freshness values that were stored up to this point. The tolerance value refers to the value for allowing reduction of the freshness value by taking into account the out of sequence. For details, refer to clause A.5.3.3.

Prosedure 1:

The receiver ECU checks its stored reference values and tolerance values for the list of previously-received freshness values against the freshness value (received value) of the Secured I-PDU to be verified, and perform the following procedures according to the comparison result.

- When "reference value < received value",

  the verification of freshness value is successful.

- When "tolerance value $\leq$ received value $\leq$ reference value",

  proceed to the Prosedure 2.

- When "received value < tolerance value",

  the verification of freshness value fails. The receiver ECU stops the verification and drop the Secured I-PDU.

Prosedure 2:

The receiver ECU checks its stored the list of previously-received freshness values against the freshness value (received value) of the Secured I-PDU to be verified, and perform the following procedures according to the comparison result.

- When the received value is not found in the list of previously-received freshness values,

  the verification of freshness value is successful.

- When the received value is found in the list of previously-received freshness values,

  the verification of freshness value fails. The receiver ECU stops the verification and drop the Secured I-PDU.

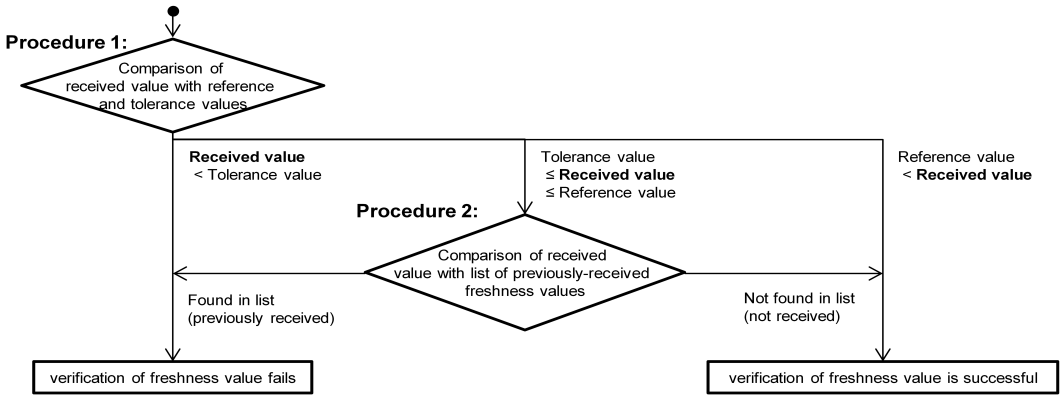

Figure A.12: Verification of freshness value

A.5.3.3 Successful verification of I-PDUs

When SecOC sends a notification of the verification result (verification is successful), the receiver ECU maintains the constructed freshness value for verification as the previously received value.

In case that the alternative method, which is to prevent discarding if out of sequence, is used, refers to the following specifications. There are two ways about list of previously-received freshness values.

Specification 1:

When SecOC sends a notification of the verification result (verification is successful), the receiver ECU stores the freshness values used for MAC verification (trip counter and message counter) and updates the list of previously-received freshness values corresponding to the `SecOCFreshnessValueId`.

The freshness values that were stored up to this point are included in the list of previously-received freshness values up to a quantity of ReceivedFreshnessValueList-Size by counting from the largest value. Among the values in the list of previously-received freshness values, the largest value is used as the reference value, and the smallest value is used as the tolerance value. The list of previously-received freshness values does not store duplicate freshness values.

Specification 2:

When SecOC sends a notification of the verification result (verification is successful), the receiver ECU stores the freshness values used for MAC verification (trip counter and message counter) and updates the list of previously-received freshness values corresponding to the SecOCFreshnessValueId.

Freshness values stored up to this point that are the tolerance values or more and reference value or less are included in the list of previously-received freshness values. Among the freshness values that were stored up to this point, the largest value is used as the reference value, and the "reference value - FreshnessValueToleranceWindow" is used as the tolerance value. The list of previously-received freshness values does not store duplicate freshness values.

Note:

ReceivedFreshnessValueListSize is number of freshness values stored as previously-received freshness values and its value range is from 1 to 255. FreshnessValueToleranceWindow is value for allowing reduction in freshness value and its value range is from 0 to 255. In principle, for the same freshness value occurs, it is treated as an error message in "Freshness value comparison", but cases where the same freshness value is handled could occur when the message count is at the maximum value.

Note:

When the trip counter is incremented, the application may store the incremented value to the non-volatile memory. It is preferable that the value is stored securely. However, storing the failsafe value into the non-volatile memory should not be implemented.

Even when the trip counter changes from the maximum value to the initial value (see clause A.5.1.2), it is treated as an increment and is stored in the non-volatile memory.

B Not applicable requirements

[SWS_SecOC_NA_00999]

Upstream requirements: SRS_BSW_00004, SRS_BSW_00167, SRS_BSW_00168, SRS_BSW_-00170, SRS_BSW_00336, SRS_BSW_00339, SRS_BSW_00375, SRS_BSW_00380, SRS_BSW_00383, SRS_BSW_00388, SRS_BSW_-00389, SRS_BSW_00390, SRS_BSW_00392, SRS_BSW_00393, SRS_BSW_00394, SRS_BSW_00395, SRS_BSW_00396, SRS_BSW_-00397, SRS_BSW_00398, SRS_BSW_00399, SRS_BSW_00400, SRS_BSW_00405, SRS_BSW_00406, SRS_BSW_00409, SRS_BSW_-00416, SRS_BSW_00417, SRS_BSW_00419, SRS_BSW_00422, SRS_BSW_00423, SRS_BSW_00424, SRS_BSW_00427, SRS_BSW_-00428, SRS_BSW_00429, SRS_BSW_00433, SRS_BSW_00437, SRS_BSW_00438, SRS_BSW_00451, SRS_BSW_00452, SRS_BSW_-00458, SRS_BSW_00461, SRS_BSW_00466, SRS_BSW_00467, SRS_BSW_00469, SRS_BSW_00470, SRS_BSW_00471, SRS_BSW_-00472

⌈

These requirements are not applicable to this specification.⌋

C Mentioned Class Tables

Class	**CryptoServicePrimitive**			
Package	M2::AUTOSARTemplates::SystemTemplate::SecureCommunication			
Note	This meta-class has the ability to represent a crypto primitive. **Tags:** atp.recommendedPackage=CryptoPrimitives			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
algorithmFamily	String	0..1	attr	This attribute represents a description of the family (e.g. AES) of crypto algorithm implemented by the crypto primitive.
algorithmMode	String	0..1	attr	This attribute represents a description of the mode of the crypto algorithm implemented by the crypto primitive.
algorithm Secondary Family	String	0..1	attr	This attribute represents a further description of the secondary family of crypto algorithm implemented by the crypto primitive. The secondary family is needed for the specification of the hash algorithm for a signature check, e.g. using RSA.

Table C.1: CryptoServicePrimitive

Class	**SecureCommunicationAuthenticationProps**			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::FibexCore::CoreCommunication			
Note	Authentication properties used to configure SecuredIPdus.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	SecureCommunicationPropsSet.authenticationProps			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
authInfoTx Length	PositiveInteger	0..1	attr	This attribute defines the length in bits of the authentication code to be included in the payload of the authenticated Pdu.

Table C.2: SecureCommunicationAuthenticationProps

Class	**SecureCommunicationFreshnessProps**			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::FibexCore::CoreCommunication			
Note	Freshness properties used to configure SecuredIPdus.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	SecureCommunicationPropsSet.freshnessProps			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
freshness CounterSync Attempts	PositiveInteger	0..1	attr	This attribute defines the number of Freshness Counter re-synchronization attempts when a verification failed for a Secured I-PDU. If the value is zero, there will be no additional verification attempt to synchronize with a potentially better fitting Freshness Counter value. This attribute is only applicable if useFreshnessTimestamp is FALSE.

▽

△

Class	**SecureCommunicationFreshnessProps**			
freshness TimestampTime PeriodFactor	PositiveInteger	0..1	attr	This attribute defines a factor that specifies the time period for the Freshness Timestamp. It holds a multiplication factor that specifies the concrete meaning of a Freshness Timestamp increment by one on basis of microseconds.
freshnessValue Length	PositiveInteger	0..1	attr	This attribute defines the complete length in bits of the Freshness Value. As long as the key doesn't change the counter shall not overflow. The length of the counter shall be determined based on the expected life time of the corresponding key and frequency of usage of the counter.
freshnessValue TxLength	PositiveInteger	0..1	attr	This attribute defines the length in bits of the Freshness Value to be included in the payload of the Secured I-PDU. This length is specific to the least significant bits of the complete Freshness Counter. If the attribute is 0 no Freshness Value is included in the Secured I-PDU.
useFreshness Timestamp	Boolean	0..1	attr	This attribute specifies whether the Freshness Value is generated through individual Freshness Counters or by a Timestamps. The value is set to TRUE when Timestamps are used.

Table C.3: SecureCommunicationFreshnessProps