

<b>Document Title</b>	Specification of Secure Onboard Communication Protocol
<b>Document Owner</b>	AUTOSAR
<b>Document Responsibility</b>	AUTOSAR
<b>Document Identification No</b>	969

<b>Document Status</b>	published
<b>Part of AUTOSAR Standard</b>	Foundation
<b>Part of Standard Release</b>	R21-11

<b>Document Change History</b>			
<b>Date</b>	<b>Release</b>	<b>Changed by</b>	<b>Description</b>
2021-11-25	R21-11	AUTOSAR Release Management	<ul style="list-style-type: none"> <li>no content changes</li> </ul>
2020-11-30	R20-11	AUTOSAR Release Management	<ul style="list-style-type: none"> <li>Initial release</li> </ul>

## **Disclaimer**

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

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

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

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

The word AUTOSAR and the AUTOSAR logo are registered trademarks.

## Table of Contents

1	Introduction and overview	4
1.1	Protocol purpose and objectives	4
1.2	Applicability of the protocol	5
1.2.1	Constraints and assumptions	5
1.2.2	Limitations	5
1.2.3	Adaptation in case of asymmetric approach	5
1.3	Dependencies	6
1.3.1	Dependencies to other protocol layers	6
1.3.2	Dependencies to other standards and norms	6
1.3.3	Dependencies to the Application Layer	6
2	Use Cases	6
3	Protocol Requirements	7
3.1	Requirements Traceability	7
4	Definition of terms and acronyms	8
4.1	Acronyms and abbreviations	8
4.2	Definition of terms	8
5	Protocol specification	10
5.1	Specification of the security solution	10
5.1.1	Basic entities of the security solution	11
5.1.2	Authentication of I-PDUs	14
5.1.3	Verification of I-PDUs	17
5.1.3.1	Successful verification of I-PDUs	20
5.1.3.2	Skipping Authentication for Secured I-PDUs at SecOC	21
5.1.3.3	Error handling and discarding of reception	21
5.2	Initialization	21
5.3	Error detection	22
5.4	Security Profiles	22
5.4.1	Overview of security profiles	22
5.4.2	SecOC Profile 1 (or 24Bit-CMAC-8Bit-FV)	22
5.4.3	SecOC Profile 2 (or 24Bit-CMAC-No-FV)	23
5.4.4	SecOC Profile 3 (or JASPAR)	23
6	Configuration parameters	23
7	Protocol usage and guidelines	27

# 1 Introduction and overview

**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 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.

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 `Authentic 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`.

## 1.1 Protocol purpose and objectives

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.

## 1.2 Applicability of the protocol

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

### 1.2.1 Constraints and assumptions

This chapter has no content.

### 1.2.2 Limitations

The protocol specification aims to ensure compatibility between AP and CP, and it assumes the communication is realized over ethernet.

Depending of the communication paradigm between AP and CP, the functionality of the protocol is limited. In the case of SOME/IP, the protocol will not support separate transmission of Authentic PDU and Cryptographic PDU and will not support usage of part of the payload as freshness information. (the details are described in the chapter Configuration Parameters.)

### 1.2.3 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 must 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.

## 1.3 Dependencies

### 1.3.1 Dependencies to other protocol layers

The interaction of SecOC with the lower layer of the communication stack will depend on the on the platform architecture (AP or CP), and in the case of a CP implementation, it will also depend on the type of transmission: direct transmission, triggered transmission or transport protocol. These design specific dependencies are not part of the protocol specification.

### 1.3.2 Dependencies to other standards and norms

[1] *IEC 7498-1 The Basic Model, IEC Norm, 1994*

[2] *National Institute of Standards and Technology (NIST): FIPS-180-4, Secure Hash Standard (SHS), March 2012, available electronically at <http://csrc.nist.gov/publications/fips/fips180-4/fips-180-4.pdf>*

[3] *FIPS Pub 197: Advanced Encryption Standard (AES), U.S. Department of Commerce, Information Technology Laboratory (ITL), National Institute of Standards and Technology (NIST), Gaithersburg, MD, USA, Federal Information Processing Standards Publication, 2001, electronically available at <http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf>*

### 1.3.3 Dependencies to the Application Layer

The SecOC protocol does not have dependencies to typical Automotive application. However, it relies on the existence of a software component that provides a freshness information. In addition, there could also be specialized applications that trigger a modification in SecOC behavior (e.g. for development porpoise) or applications that monitor the verification results.

## 2 Use Cases

<b>ID</b>	<b>Name</b>	<b>Description</b>
<b>UC_001</b>	SecOC SOME/IP	Secure communication between AP and CP using SOME/IP
<b>UC_002</b>	SecOC SignalBased	Secure communication between AP and CP using signal-based communication and SignalToService translation

### 3 Protocol Requirements

#### 3.1 Requirements Traceability

Requirement	Description	Satisfied by
[RS_Main_00510]	AUTOSAR shall support secure onboard communication	<a href="#">[PRS_SecOc_00101]</a> <a href="#">[PRS_SecOc_00102]</a> <a href="#">[PRS_SecOc_00103]</a> <a href="#">[PRS_SecOc_00104]</a> <a href="#">[PRS_SecOc_00105]</a> <a href="#">[PRS_SecOc_00200]</a> <a href="#">[PRS_SecOc_00201]</a> <a href="#">[PRS_SecOc_00202]</a> <a href="#">[PRS_SecOc_00203]</a> <a href="#">[PRS_SecOc_00204]</a> <a href="#">[PRS_SecOc_00205]</a> <a href="#">[PRS_SecOc_00206]</a> <a href="#">[PRS_SecOc_00207]</a> <a href="#">[PRS_SecOc_00208]</a> <a href="#">[PRS_SecOc_00209]</a> <a href="#">[PRS_SecOc_00210]</a> <a href="#">[PRS_SecOc_00211]</a> <a href="#">[PRS_SecOc_00212]</a> <a href="#">[PRS_SecOc_00213]</a> <a href="#">[PRS_SecOc_00214]</a> <a href="#">[PRS_SecOc_00215]</a> <a href="#">[PRS_SecOc_00216]</a> <a href="#">[PRS_SecOc_00217]</a> <a href="#">[PRS_SecOc_00218]</a> <a href="#">[PRS_SecOc_00220]</a> <a href="#">[PRS_SecOc_00300]</a> <a href="#">[PRS_SecOc_00301]</a> <a href="#">[PRS_SecOc_00302]</a> <a href="#">[PRS_SecOc_00303]</a> <a href="#">[PRS_SecOc_00304]</a> <a href="#">[PRS_SecOc_00305]</a> <a href="#">[PRS_SecOc_00306]</a> <a href="#">[PRS_SecOc_00307]</a> <a href="#">[PRS_SecOc_00309]</a> <a href="#">[PRS_SecOc_00310]</a> <a href="#">[PRS_SecOc_00311]</a> <a href="#">[PRS_SecOc_00312]</a> <a href="#">[PRS_SecOc_00313]</a> <a href="#">[PRS_SecOc_00314]</a> <a href="#">[PRS_SecOc_00315]</a> <a href="#">[PRS_SecOc_00316]</a> <a href="#">[PRS_SecOc_00317]</a> <a href="#">[PRS_SecOc_00318]</a> <a href="#">[PRS_SecOc_00320]</a> <a href="#">[PRS_SecOc_00341]</a> <a href="#">[PRS_SecOc_00342]</a> <a href="#">[PRS_SecOc_00400]</a> <a href="#">[PRS_SecOc_00600]</a>

Requirement	Description	Satisfied by
		[PRs_SecOc_00610] [PRs_SecOc_00620] [PRs_SecOc_00630]
[SRS_BSW_00101]	The Basic Software Module shall be able to initialize variables and hardware in a separate initialization function	[PRs_SecOc_00401]
[SRS_BSW_00171]	Optional functionality of a Basic-SW component that is not required in the ECU shall be configurable at pre-compile-time	[PRs_SecOc_00100]
[SRS_BSW_00337]	Classification of development errors	[PRs_SecOc_00500]
[SRS_BSW_00350]	All AUTOSAR Basic Software Modules shall allow the enabling/disabling of detection and reporting of development errors.	[PRs_SecOc_00500]
[SRS_BSW_00385]	List possible error notifications	[PRs_SecOc_00330] [PRs_SecOc_00340] [PRs_SecOc_00341] [PRs_SecOc_00500]
[SRS_BSW_00426]	BSW Modules shall ensure data consistency of data which is shared between BSW modules	[PRs_SecOc_00219]
[SRS_BSW_00450]	A Main function of a un-initialized module shall return immediately	[PRs_SecOc_00500]

## 4 Definition of terms and acronyms

### 4.1 Acronyms and abbreviations

Abbreviation / Acronym:	Description:
SecOC	Secure Onboard Communication
MAC	Message Authentication Code
FV	Freshness Value
FM	Freshness Manager

### 4.2 Definition of terms

Terms:	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.

Terms:	Description:
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.
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).

Terms:	Description:
Entity authentication	<p>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).</p> <p><b>Note:</b> 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.</p> <p>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.</p>
Message authentication	<p>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).</p>
Secured I-PDU	<p>A Secured I-PDU is an AUTOSAR I-PDU that contains Payload of an Authentic I-PDU supplemented by additional Authentication Information.</p>
Transaction authentication	<p>Transaction authentication denotes message authentication augmented to additionally provide uniqueness and timeliness guarantees on data (thus preventing undetectable message replay).</p>

## 5 Protocol specification

### 5.1 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.

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 **Authentic 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**.

### 5.1.1 Basic entities of the security solution

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 [Figure 5.1](#).

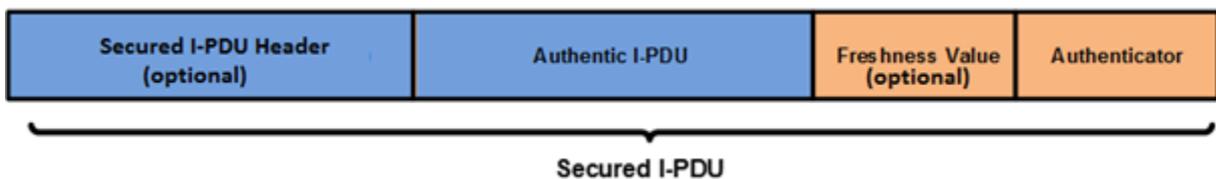


Figure 5.1: **Secured I-PDU** contents

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.

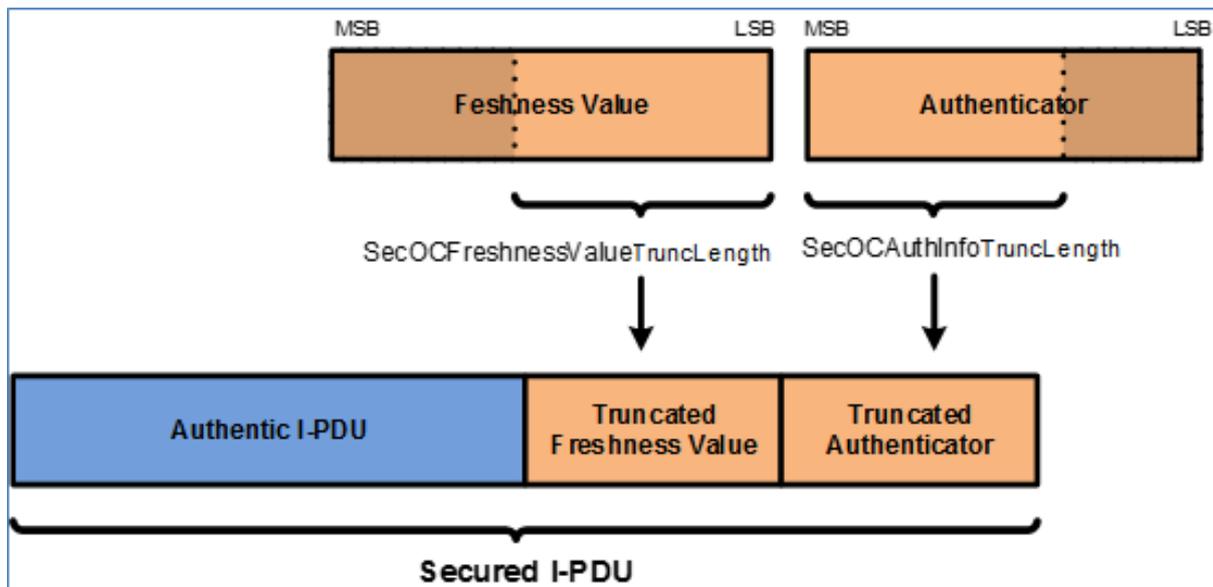
The **Authenticator** (e.g. MAC) refers to a unique authentication data string generated using a Key, Data Identifier of the **Secured I-PDU**, Authentic Payload, and Freshness Value. The **Authenticator** provides a high level of confidence that the data in an **Authentic I-PDU** is generated by a legitimate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

Depending on the authentication algorithm 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-PDU*s may have different *Authenticator* lengths included in the payload of the *Secured 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 5.2** shows an example of the truncation of the *Authenticator* and the Freshness Values respecting the parameter *SecOCFreshnessValueTruncLength* and *SecOCAuthInfoTruncLength*.



**Figure 5.2:** 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 [4] based on AES [3] 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 bits, 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 [4]. 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.

**[PRS\_SecOc\_00100]** [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.]([SRS\\_BSW\\_00171](#))

**[PRS\_SecOc\_00101]** [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.]([RS\\_Main\\_00510](#))

**[PRS\_SecOc\_00102]** [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`.]([RS\\_Main\\_00510](#))

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.

**[PRS\_SecOc\_00103]** [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`.]([RS\\_Main\\_00510](#))

**[PRS\_SecOc\_00104]** [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.]([RS\\_Main\\_00510](#))

**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.

**[PRS\_SecOc\_00105]** [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.]([RS\\_Main\\_00510](#))

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

### 5.1.2 Authentication of I-PDUs

**[PRS\_SecOc\_00200]** [The creation of a `Secured I-PDU` and thus the authentication of an `Authentic I-PDU` consists 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`

]([RS\\_Main\\_00510](#))

**[PRS\_SecOc\_00201]** [Whenever the `DataToAuthenticator` is constructed for a specific PDU, `SecOC` calls `Freshness Manager` to get a truncated freshness value if `SecOCProvideTxTruncatedFreshnessValue` is set to `TRUE` or a full (not truncated) freshness value if `SecOCProvideTxTruncatedFreshnessValue` is set to `FALSE`.]([RS\\_Main\\_00510](#))

**[PRS\_SecOc\_00202]** [For every transmission request that is queued to `SecOC` an authentication build counter shall be maintained.]([RS\\_Main\\_00510](#))

**[PRS\_SecOc\_00203]** [Upon the initial processing of a transmission request of a `Secured I-PDU` `SecOC` shall set the authentication build counter to 0.]([RS\\_Main\\_00510](#))

**[PRS\_SecOc\_00204]** [If for the transmitted PDU either the query of the freshness value or the calculation of the authenticator does not return a result (example: a new freshness value could not be provided, or a crypto stack is busy) the authentication build counter shall be incremented.]([RS\\_Main\\_00510](#))

**[PRS\_SecOc\_00205]** [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.]([RS\\_Main\\_00510](#))

**[PRS\_SecOc\_00206]** [If the authentication build counter has reached the configuration value `SecOCAuthenticationBuildAttempts`, or the query of the freshness

function has returns a non-recoverable error (example: a systematic failure due to freshness value configuration) or the calculation of the authenticator has returns a non-recoverable error (example: a systematic error due to a not configured key), the `SecOC` module shall use `SecOCDefaultAuthenticationInformationPattern` for all the bytes of Freshness Value and Authenticator to build the `Authentication Information` if sending `SecOCDefaultAuthenticationInformationPattern` is enabled. If sending `SecOCDefaultAuthenticationInformationPattern` is not enabled, the `SecOC` module shall remove the `Authentic I-PDU` from its internal buffer and cancel the transmission request. *(RS\_Main\_00510)*

**Note:**

Example:

`SecOCFreshnessValueTxLength = 4bits`

`SecOCAuthInfoTxLength = 20 bits`

`SecOCDefaultAuthenticationInformationPattern = 0xA5`

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

**[PRS\_SecOc\_00207]** [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.] *(RS\_Main\_00510)*

**[PRS\_SecOc\_00208]** [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.] *(RS\_Main\_00510)*

**[PRS\_SecOc\_00209]** [The `SecOC` module shall generate the `Authenticator` by passing `DataToAuthenticator`, `length of DataToAuthenticator` into the Authentication Algorithm corresponding to the configuration of cryptographic service configured.] *(RS\_Main\_00510)*

**[PRS\_SecOc\_00210]** [The `SecOC` module shall truncate the resulting `Authenticator` down to the number of bits specified by `SecOCAuthInfoTruncLength`.] *(RS\_Main\_00510)*

**[PRS\_SecOc\_00211]** [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`] *(RS\_Main\_00510)*

**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](#).

**[PRS\_SecOc\_00212]** [The [SecOC](#) module shall copy the complete [Authentic I-PDU](#) to its internal memory before starting transmission of the corresponding [Secured I-PDU](#).] ([RS\\_Main\\_00510](#))

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

**[PRS\_SecOc\_00213]** [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](#).] ([RS\\_Main\\_00510](#))

**[PRS\_SecOc\_00214]** [[SecOC](#) shall transmit an [Authentic I-PDU](#) and its corresponding [Cryptographic I-PDU](#) within the same main function cycle.] ([RS\\_Main\\_00510](#))

**[PRS\_SecOc\_00215]** [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](#)] ([RS\\_Main\\_00510](#))

**Note:** "|" denotes concatenation

**[PRS\_SecOc\_00216]** [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](#).] ([RS\\_Main\\_00510](#))

**[PRS\_SecOc\_00217]** [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 [\[PRS\\_SecOc\\_00200\]](#).] ([RS\\_Main\\_00510](#))

**[PRS\_SecOc\_00218]** [The [SecOC](#) module shall provide separate buffers for the [Authentic I-PDU](#) and the [Secured I-PDU](#).] ([RS\\_Main\\_00510](#))

**[PRS\_SecOc\_00219]** [Any transmission request from the upper layer interfaces of the communication stack shall overwrite the buffer that contains the [Authentic I-PDU](#) without affecting the buffer of the respective [Secured I-PDU](#).] ([SRS\\_BSW\\_00426](#))

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.

**[PRS\_SecOc\_00220]** [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](#).] ([RS\\_Main\\_00510](#))

### 5.1.3 Verification of I-PDUs

**[PRS\_SecOc\_00300]** [The verification of a [Secured I-PDU](#) consists of the following six steps:

1. Parse [Authentic I-PDU](#), Freshness Value and [Authenticator](#)
2. Get Freshness Value from Freshness Manager
3. Construct Data to Authentication
4. Verify [Authentication Information](#)
5. Send Confirmation to Freshness Manager
6. Pass [Authentic I-PDU](#) to upper layer

]([RS\\_Main\\_00510](#))

**[PRS\_SecOc\_00301]** [For every processed [Secured I-PDU](#) within SecOC an authentication build counter and an authentication verify attempt counter shall be maintained.]([RS\\_Main\\_00510](#))

**[PRS\_SecOc\_00302]** [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.]([RS\\_Main\\_00510](#))

**[PRS\_SecOc\_00303]** [If the query of the freshness value for the received PDU does not return a result (example: a new freshness value could not be provided) the authentication build counter shall be incremented and no attempt for verification of authentication shall be executed.]([RS\\_Main\\_00510](#))

**[PRS\_SecOc\_00304]** [If the verification of the authenticator returns a recoverable error (example: crypto stack is busy), the authentication build counter shall be incremented.]([RS\\_Main\\_00510](#))

**[PRS\_SecOc\_00305]** [If the authentication build attempts have failed and the authentication build counter has not yet reached the configuration value , the freshness attempt and the [Authenticator](#) verification shall be retried in the next call to the Rx main function.]([RS\\_Main\\_00510](#))

**[PRS\_SecOc\_00306]** [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.]([RS\\_Main\\_00510](#))

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

**[PRS\_SecOc\_00307]** [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 `VerificationResultType` shall be set to `SECOC_AUTHENTICATIONBUILDFAILURE`.

If `SecOC_VerifyStatusOverride` is used, the verification result and I-PDU are handled according to `overrideStatus` value. *(RS\_Main\_00510)*

**[PRS\_SecOc\_00308]** [If the query of the freshness function returns a non-recoverable error (example: a systematic failure due to freshness value configuration ) the `SecOC` module shall remove the `Authentic I-PDU` from its internal buffer and shall drop the received message. The `VerificationResultType` shall be set to `SECOC_FRESHNESSFAILURE`.] *()*

**[PRS\_SecOc\_00309]** [If the authentication verify attempt counter has reached the configuration value `SecOCAuthenticationVerifyAttempts` or the verification of the `Authenticator` has returned a non-recoverable error (example: a systematic error due to a not configured key) the `SecOC` module shall remove the `Authentic I-PDU` from its internal buffer and shall drop the received message. The `VerificationResultType` shall be set to `SECOC_VERIFICATIONFAILURE`. If `SecOC_VerifyStatusOverride` is used, the verification result and I-PDU are handled according to `overrideStatus` value.] *(RS\_Main\_00510)*

**[PRS\_SecOc\_00310]** [If the verification of the `Authenticator` was successful, the `VerificationResultType` shall be set to `SECOC_VERIFICATIONSUCCESS`.] *(RS\_Main\_00510)*

**[PRS\_SecOc\_00311]** [The Freshness Management shall use the verification status callout function 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.] *(RS\_Main\_00510)*

**Note:** `SecOC` allows to overwrite the status. 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.

**[PRS\_SecOc\_00312]** [If the Rx freshness request function returns a non-recoverable error (example: a systematic failure due to freshness value configuration) 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_RE_FRESHNESS_FAILURE` shall be reported.] *(RS\_Main\_00510)*

**[PRS\_SecOc\_00313]** [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 `Secured I-PDU`. If `SecOC_VerifyStatusOverride` is used, the verification result and I-PDU are handled according to `overrideStatus` value.] *(RS\_Main\_00510)*

**Note:** This applies to all instances when a Secured I-PDU is received by SecOC from the lower layer, 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.

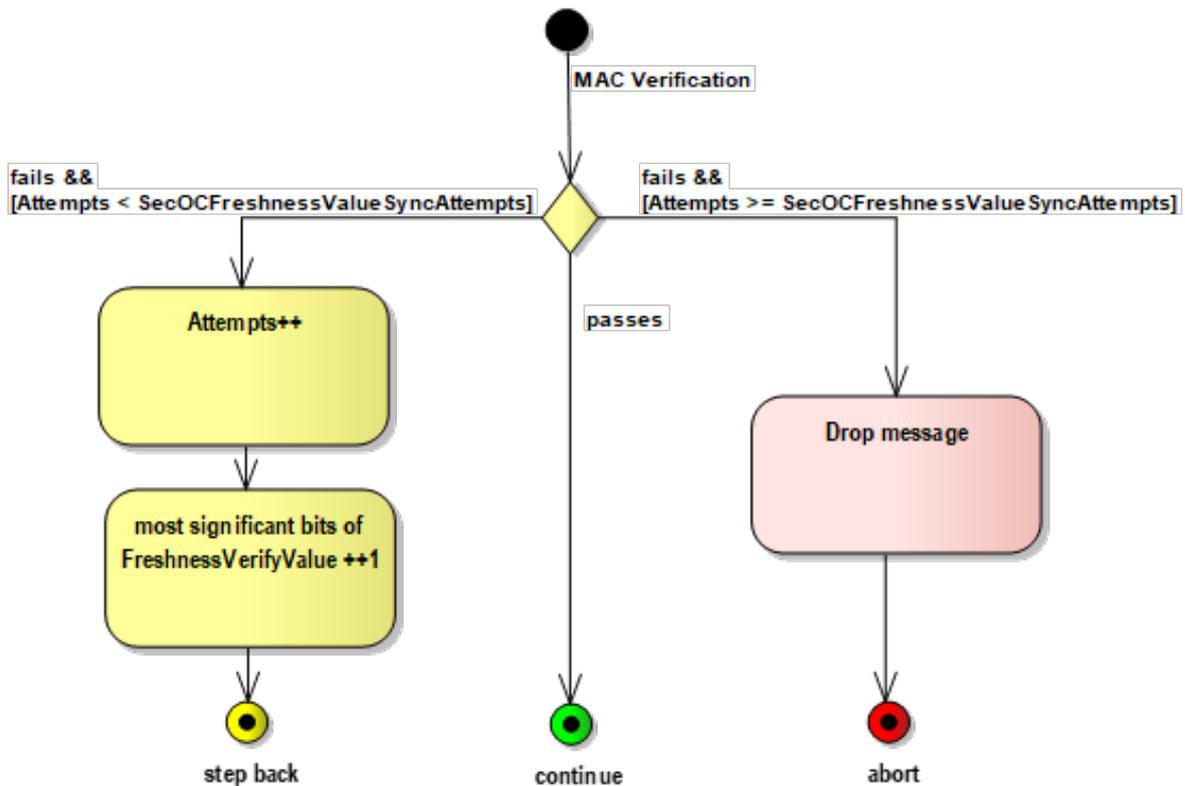
**[PRS\_SecOc\_00314]** [If SecOCRxSecuredPduCollection is used then SecOC shall not attempt to verify the Secured I-PDU until it has received and buffered an Authentic I-PDU and Cryptographic I-PDU with matching Message Linker values. If SecOC\_VerifyStatusOverride is used, the verification result and I-PDU are handled according to overrideStatus value.](RS\_Main\_00510)

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

**[PRS\_SecOc\_00315]** [Upon reception of a Secured I-PDU, SecOC shall parse the Authentic I-PDU, the Freshness Value and the Authenticator from it.](RS\_Main\_00510)

**[PRS\_SecOc\_00316]** [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.](RS\_Main\_00510)

**[PRS\_SecOc\_00317]** [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 configured cryptographic service. If SecOC\_VerifyStatusOverride is used, the verification result and I-PDU are handled according to overrideStatus value.](RS\_Main\_00510)



**Figure 5.3: Verification of MAC**

**[PRS\_SecOc\_00318]** [The `SecOC` module shall report each individual verification status (the final one as well as all intermediate ones) according to its current configuration (see parameter `SecOCVerificationStatusPropagationMode`).] ([RS\\_Main\\_00510](#))

**Note:** If the Freshness Manager requires the status of a `Secured I-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`.

### 5.1.3.1 Successful verification of I-PDUs

**[PRS\_SecOc\_00320]** [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 communication stack.] ([RS\\_Main\\_00510](#))

### 5.1.3.2 Skipping Authentication for Secured I-PDUs at SecOC

**[PRS\_SecOc\_00330]** [For a Rx *Secured I-PDU*, there should be a configuration option to skip the verification. In this case the *SecOC* module shall extract the *Authentic I-PDU* without Authentication.] (*SRS\_BSW\_00385*)

### 5.1.3.3 Error handling and discarding of reception

**[PRS\_SecOc\_00340]** [If the lower layer transport protocol module reports an error during reception of a *Secured I-PDU*, the *SecOC* module shall drop the *Secured I-PDU* and free all corresponding buffers.] (*SRS\_BSW\_00385*)

**[PRS\_SecOc\_00341]** [If the Crypto module reports an error during verification (verification cannot be performed) 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.] (*RS\_Main\_00510*, *SRS\_BSW\_00385*)

**[PRS\_SecOc\_00342]** [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.] (*RS\_Main\_00510*)

**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.

## 5.2 Initialization

The *SecOC* module provides an initialization function. 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 the initialization function before calling any other function of the *SecOC* module except function that return information about *SecOC* implementation (example: version information). The implementer has to take care of errors that might be returned by calling *SecOC* functions before initialization.

**[PRS\_SecOc\_00400]** [Within *SecOC* initialization function, the module shall initialize all internal global variables and the buffers of the *SecOC I-PDUs*.] (*RS\_Main\_00510*)

**[PRS\_SecOc\_00401]** [The AUTOSAR *SecOC* module shall fill not used areas of a transmitted *Secured PDU* with a value determined by configuration parameter *SecOC-TxPduUnusedAreasDefault* e.g. 0xFF.] (*SRS\_BSW\_00101*)

## 5.3 Error detection

**[PRS\_SecOc\_00500]** [The `SecOC` module shall be able to report errors in case the module is called before initialization, in case a freshness value cannot be provided or in case there is no cryptographic operation configured for the verification check.]([SRS\\_BSW\\_00337](#), [SRS\\_BSW\\_00350](#), [SRS\\_BSW\\_00385](#), [SRS\\_BSW\\_00450](#))

## 5.4 Security Profiles

### 5.4.1 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.

**[PRS\_SecOc\_00600]** [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.]([RS\\_Main\\_00510](#))

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

**[PRS\_SecOc\_00610]** [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`.]([RS\\_Main\\_00510](#))

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

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

**[PRS\_SecOc\_00620]** [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.]([RS\\_Main\\_00510](#))

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

### 5.4.4 SecOC Profile 3 (or JASPAR)

**[PRS\_SecOc\_00630]** [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.]([RS\\_Main\\_00510](#))

Parameter	Configuration value
The algorithm for the MAC (parameter <code>algorithmFamily</code> )	CRYPTO_ALGOFAM_AES
The algorithm mode for the MAC (parameter <code>algorithmMode</code> )	CRYPTO_ALGOMODE_CMAC
Additional algorithm family configuration (parameter <code>algorithmSecondaryFamily</code> , not used in this profile)	CRYPTO_ALGOFAM_NOT_SET
Length of Freshness Value (parameter <code>SecOCFreshnessValueLength</code> )	64 bits
Length of truncated Freshness Value (parameter <code>SecOCFreshnessValueTruncLength</code> )	4 bits
Length of truncated MAC (parameter <code>SecOCAuthInfoTruncLength</code> )	24 bits

## 6 Configuration parameters

The table below describes the configuration parameters for the protocol. For the communication between AP and CP using SOME/IP network binding or raw data streaming, the protocol has a reduced functionality, therefore some parameters are not available

in AP or they are implementation specific. These are described in column "Applicability to AP SOME/IP network binding". As a consequence, the requirements referring these parameters are not applicable either. The parameters that are not available are because following features are not available:

- It is not possible to use part of the Authentic PDU to construct as freshness information
- It is not possible to separate the Secure PDU in two different PDUs: Authentic and Cryptographic PDUs
- Provision of Freshness value already truncated by FM (the truncation is always done by SecOC)

<b>Parameter</b>	<b>Description</b>	<b>Applicability to AP SOME/IP network binding</b>
SecOCAuthPduHeaderLength	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.	no
SecOCFreshnessValueTruncLength	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.	yes
SecOCUseAuthDataFreshness	A Boolean value that indicates if a part of the Authentic I-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 I-PDU.	no
SecOCAuthDataFreshnessStartPosition	This value determines the start position in bits (uint16) of the Authentic PDU that shall be passed on to the Freshness SWC.	no





SecOCAuthDataFreshnessLen	This attribute defines the length in bits of the authentic PDU data that is passed to the SWC that verifies and generates the Freshness.	no
SecOCProvideTxTruncatedFreshnessValue	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.	no
SecOCAuthenticationBuildAttempts	Parameter specifies the number of authentication build attempts.	Implementation specific
SecOCDefaultAuthenticationInformationPattern	The parameter describes the behavior of SecOC when authentication build counter has reached the configuration value <a href="#">SecOCAuthenticationBuildAttempts</a> , or the query of the freshness function returns non-recoverable error (example: a systematic failure due to freshness value configuration) or the calculation of the <a href="#">Authenticator</a> has returned a non-recoverable error (example: a systematic failure due wrong crypto configuration or missing key). If the configuration parameter is not present, SecOC module shall remove the <a href="#">Authentic I-PDU</a> 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 <a href="#">Authenticator</a> when building the Authentication Information, and will not cancel the transmission request.	Implementation specific





SecOCAuthenticationVerifyAttempts	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 <a href="#">Secured I-PDU</a> . If zero is set, then only one authentication verification attempt is done.	Implementation specific
SecOCDataId	This parameter defines a unique numerical identifier for the <a href="#">Secured I-PDU</a> .	yes
SecOCFreshnessValueId	This parameter defines the Id of the Freshness Value	yes
SecOCTxAuthServiceConfigRef	This reference is used to define which crypto service function is called for authentication	Implementation specific
SecOCAuthInfoTruncLength	This parameter defines the length in bits of the authentication code to be included in the payload of the <a href="#">Secured I-PDU</a> .	yes
SecOC_VerifyStatusOverride	When this configuration option is set to TRUE then the functionality inside the function <a href="#">SecOC_VerifyStatusOverride</a> to send I-PDUs to upper layer independent of the verification result is enabled.	Implementation specific
SecOCTxSecuredPduCollection	Two separate Pdus are transmitted to the lower layer: <a href="#">Authentic I-PDU</a> and <a href="#">Cryptographic I-PDU</a> .	no
SecOCFreshnessValueLength	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 lifetime of the corresponding key and frequency of usage of the counter.	yes





SecOCTxPduUnusedAreasDefault	The AUTOSAR <code>SecOC</code> module fills not used areas of a transmitted Secured Pdu with this byte pattern. This attribute is mandatory to avoid undefined behavior.	Implementation specific
SecOCUseMessageLink	<code>SecOC</code> links an <code>Authentic I-PDU</code> and Cryptographic I-PDU together by repeating a specific part (Message Linker) of the <code>Authentic I-PDU</code> in the Cryptographic I-PDU.	no
SecOCMessageLinkPos	The position of the Message Linker inside the <code>Authentic I-PDU</code> in bits. The bit counting is done according to 01068 and the bit ordering is done according to TPS_SYST_01069.	no
SecOCMessageLinkLen	Length of the Message Linker inside the <code>Authentic I-PDU</code> in bits.	no
SecOCSecuredRxPduVerification	This parameter defines whether the signature authentication or MAC verification shall be performed on this <code>Secured I-PDU</code> . If set to false, the <code>SecOC</code> module extracts the <code>Authentic I-PDU</code> from the <code>Secured I-PDU</code> without verification.	no

## 7 Protocol usage and guidelines

This chapter has no content.

## References

- [1] IEC: The Basic Model, IEC Norm
- [2] NIST: Secure Hash Standard (SHS)  
<http://csrc.nist.gov/publications/fips/fips180-4/fips-180-4.pdf>
- [3] NIST: Announcing the Advanced Encryption Standard (AES)  
<http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf>

- [4] 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](http://csrc.nist.gov/publications/nistpubs/800-38B/SP_800-38B.pdf)