

<b>Document Title</b>	Requirements on Crypto Stack
<b>Document Owner</b>	AUTOSAR
<b>Document Responsibility</b>	AUTOSAR
<b>Document Identification No</b>	426

<b>Document Status</b>	published
<b>Part of AUTOSAR Standard</b>	Classic Platform
<b>Part of Standard Release</b>	R25-11

<b>Document Change History</b>			
<b>Date</b>	<b>Release</b>	<b>Changed by</b>	<b>Description</b>
2025-11-27	R25-11	AUTOSAR Release Management	<ul style="list-style-type: none"> <li>• No content changes</li> </ul>
2024-11-27	R24-11	AUTOSAR Release Management	<ul style="list-style-type: none"> <li>• No content changes</li> </ul>
2023-11-23	R23-11	AUTOSAR Release Management	<ul style="list-style-type: none"> <li>• No content changes</li> </ul>
2022-11-24	R22-11	AUTOSAR Release Management	<ul style="list-style-type: none"> <li>• Add support fo certificate formats CVC and X.509</li> </ul>
2021-11-25	R21-11	AUTOSAR Release Management	<ul style="list-style-type: none"> <li>• Editorial changes</li> </ul>
2020-11-30	R20-11	AUTOSAR Release Management	<ul style="list-style-type: none"> <li>• Updated supported algorithms</li> </ul>
2019-11-28	R19-11	AUTOSAR Release Management	<ul style="list-style-type: none"> <li>• Updated items in glossary and abbreviation list</li> <li>• Changed Document Status from Final to published</li> </ul>
2018-10-31	4.4.0	AUTOSAR Release Management	<ul style="list-style-type: none"> <li>• Adding Coverage of Key Manager</li> <li>• Removed Secure Counter functionality</li> <li>• Editorial changes</li> </ul>



△

2017-12-08	4.3.1	AUTOSAR Release Management	<ul style="list-style-type: none"> <li>• Default error detection renamed to development error detection</li> <li>• Editorial changes</li> </ul>
2016-11-30	4.3.0	AUTOSAR Release Management	<ul style="list-style-type: none"> <li>• Added requirements for the whole Crypto Stack and renamed the document</li> <li>• Introduced crypto job concept</li> <li>• Introduced key management concept</li> </ul>
2014-10-31	4.2.1	AUTOSAR Release Management	<ul style="list-style-type: none"> <li>• Editorial changes</li> </ul>
2013-10-31	4.1.2	AUTOSAR Release Management	<ul style="list-style-type: none"> <li>• Editorial changes</li> </ul>
2013-03-15	4.1.1	AUTOSAR Administration	<ul style="list-style-type: none"> <li>• [TPS_STDT_0078] formatting</li> <li>• Traceability of BSWAndRTE_Features</li> </ul>
2010-09-30	3.1.5	AUTOSAR Administration	<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	Scope of Document	6
2	Conventions to be used	7
3	Acronyms and abbreviations	8
4	Requirements Specification	10
4.1	Functional Overview	10
4.1.1	Supported Algorithms	10
4.2	Functional Requirements	12
4.2.1	Crypto Stack	12
4.2.1.1	General	12
4.2.1.2	Configuration	13
4.2.1.3	Initialization	14
4.2.1.4	Normal Operation	15
4.2.1.5	Shutdown Operation	20
4.2.1.6	Fault Operation	20
4.2.2	Key Manager	20
4.2.2.1	General	20
4.2.2.2	Configuration	23
4.2.2.3	Initialization	24
4.2.2.4	Normal Operation	25
4.2.2.5	Shutdown Operation	25
4.2.2.6	Fault Operation	26
4.2.3	Crypto Service Manager	26
4.2.3.1	General	26
4.2.3.2	Configuration	27
4.2.3.3	Initialization	28
4.2.3.4	Normal Operation	28
4.2.3.5	Shutdown Operation	29
4.2.3.6	Fault Operation	29
4.2.4	Crypto Interface	30
4.2.4.1	Configuration	30
4.2.4.2	Initialization	31
4.2.4.3	Normal Operation	31
4.2.4.4	Shutdown Operation	31
4.2.4.5	Fault Operation	31
4.2.5	Crypto Driver	31
4.2.5.1	Configuration	31
4.2.5.2	Initialization	32
4.2.5.3	Normal Operation	32
4.2.5.4	Shutdown Operation	32
4.2.5.5	Fault Operation	33

4.2.6 Security Event Memory	33
4.2.6.1 General	33
4.3 Non-Functional Requirements (Qualities)	34
4.3.1 General	34
4.3.2 Crypto Service Manager	34
4.3.3 Crypto Interface	36
4.3.4 Crypto Driver	36
5 References	37
A Change history of AUTOSAR traceable items	38
A.1 Traceable item history of this document according to AUTOSAR Release R25-11	38
A.1.1 Added Requirements in R25-11	38
A.1.2 Changed Requirements in R25-11	38
A.1.3 Deleted Requirements in R25-11	38

# 1 Scope of Document

This document specifies the requirements of the crypto stack:

- [1, Crypto Service Manager (Csm)]
- [2, Crypto Interface (CryIf)]
- [3, Crypto Driver (Crypto)]
- [4, Key Manager]

## 2 Conventions to be used

The representation of requirements in AUTOSAR documents follows the table specified in [TPS\_STDT\_00078], see [5, Standardization Template].

The verbal forms for the expression of obligation specified in [TPS\_STDT\_00053] shall be used to indicate requirements, see [5, Standardization Template].

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as follows.

Note that the requirement level of the document in which they are used modifies the force of these words.

- **MUST:** This word, or the adjective "LEGALLY REQUIRED", means that the definition is an absolute requirement of the specification due to legal issues.
- **MUST NOT:** This phrase, or the phrase "MUST NOT", means that the definition is an absolute prohibition of the specification due to legal issues.
- **SHALL:** This phrase, or the adjective "REQUIRED", means that the definition is an absolute requirement of the specification.
- **SHALL NOT:** This phrase means that the definition is an absolute prohibition of the specification.
- **SHOULD:** This word, or the adjective "RECOMMENDED", means that there may exist valid reasons in particular circumstances to ignore a particular item, but the full implications must be understood and carefully weighed before choosing a different course.
- **SHOULD NOT:** This phrase, or the phrase "NOT RECOMMENDED", means that there may exist valid reasons in particular circumstances when the particular behavior is acceptable or even useful, but the full implications should be understood and the case carefully weighed before implementing any behavior described with this label.
- **MAY:** This word, or the adjective "OPTIONAL", means that an item is truly optional. One vendor may choose to include the item because a particular marketplace requires it or because the vendor feels that it enhances the product while another vendor may omit the same item.

An implementation, which does not include a particular option, SHALL be prepared to interoperate with another implementation, which does include the option, though perhaps with reduced functionality. In the same vein an implementation, which does include a particular option, SHALL be prepared to interoperate with another implementation, which does not include the option (except, of course, for the feature the option provides).

### 3 Acronyms and abbreviations

The glossary below includes acronyms and abbreviations relevant to CryptoStack that are not included in the AUTOSAR Glossary [6].

<b>Abbreviation / Acronym:</b>	<b>Description:</b>
μC	Microcontroller
AES	Advanced Encryption Standard
CBC	Cipher Block Chaining
CDD	Complex Device Driver
CFB	Cipher Feedback
CMAC	Cipher-based Message Authentication Code
CPU	Central Processing Unit
CRYPTO / Crypto	Crypto Driver
CRYIF / CryIf	Crypto Interface
CSM / Csm	Crypto Service Manager
DET / Det	Default Error Tracer
ECB	Electronic Code Book
ECC	Elliptic Curve Cryptography
ECDH	Elliptic Curve Diffie-Hellman
ECDSA	Elliptic Curve Digital Signature Algorithm
ECU	Electronic Control Unit
GCM	Galois Counter Mode
GMAC	Galois-based Message Authentication Code
HMAC	Hash-based Message Authentication Code
HSM / Hsm	Hardware Security Module
HW	HardWare
KEM	Key Encapsulation Mechanism
KeyM	Key Manager
MAC	Message Authentication Code
MCAL	Micro Controller Abstraction Layer
OEM	Original Equipment Manufacturer
OFB	Output Feedback
PKI	Public Key Infrastructure
PRNG	Pseudo-Random Number Generator
RACE	Rapid Automatic Cryptographic Equipment
RAM	Random Access Memory
RIPEMD	RACE Integrity Primitives Evaluation Message Digest
RSA	Rivest-Shamir-Adleman Cryptosystem
RTE	Run Time Environment
SHA	Secure Hash Algorithm
SECOC / SecOc	Secure Onboard Communication
SW	SoftWare
SWC	SoftWare Component
SWS	SoftWare Specification
TRNG	True Random Number Generator





Vi	VendorId
XEX	Xor-Encrypt-Xor
XTS	XEX-based tweaked-codebook mode with ciphertext stealing

**Table 3.1: Acronyms and abbreviations used in the scope of this Document**

<b>Terms:</b>	<b>Description:</b>	
Crypto Driver Object	A Crypto Driver Object is an instance of a crypto module (hardware or software), which is able to perform one or more different crypto operations.	
User	A user is a configured object with an ID and configured jobs.	
Channel	A channel is the path from a Crypto Service Manager queue via the Crypto Interface to a specific Crypto Driver Object.	
Job	A job is an instance of a user's configured cryptographic primitive.	
Primitive	A primitive is an instance of a configured cryptographic algorithm realized in a Crypto Driver Object. Among others it refers to a functionality provided by the CSM to the application, the concrete underlining 'algorithmfamily' (e.g. AES, MD5, RSA, etc.), and a 'algorithmmode' (e.g. ECB, CBC, etc).	
Operation	An operation of a crypto primitive declares what part of the crypto primitive shall be performed. There are three different operations:	
	START	Operation indicates a new request of a crypto primitive, and it shall cancel all previous requests.
	UPDATE	Operation indicates, that the crypto primitive expect input data.
	FINISH	Operation indicates, that after this part all data are fed completely and the crypto primitive can finalize the calculations.
	It is also possible to perform more than one operation at once by concatenating the corresponding bits of the operation_mode argument.	
Priority	The priority of a user defines the importance of it. The higher the priority (as well in value), the more immediate the user's job will be executed. The priority of a cryptographic job is part of the user's configuration.	
Service	A service shall be understand as defined in the TR_Glossary document: A service is a type of operation that has a published specification of interface and behavior, involving a contract between the provider of the capability and the potential clients.	

**Table 3.2: Terms used in the scope of this Document**

## 4 Requirements Specification

This chapter describes all requirements driving the work to define the CryptoStack.

### 4.1 Functional Overview

The Crypto Stack offers a standardized access to cryptographic services for applications and system functions.

The cryptographic services are, e.g., the computation of hashes, the verification of asymmetrical signatures, or the symmetrical encryption of data. These services depend on underlying cryptographic primitives and cryptographic schemes. The CSM shall make it possible for different applications to use the same service but using different underlying primitives and/or schemes. E.g., one application might need to use the hash service to compute an SHA2 digest and another might need to compute an SHA1 digest. Or one application might need to verify a signature which has been computed with the RSASSA-PKCS1-V1\_5 signature scheme and using SHA1 as an underlying hash primitive, while another application might need to verify a signature computed with a different scheme which uses SHA2 as an underlying hash primitive. The Crypto Stack shall make it possible to configure which services are needed and to create several configurations for each service where schemes and primitives can be chosen.

Furthermore, since the computation of many of the cryptographic services is very computation intensive, provisions have to be made for scheduling these long computations. The jobs shall be configurable to be executed synchronously or asynchronously.

The Crypto Stack provides services with cryptography functionality, based on software libraries or on hardware modules. Also, mixed setups are possible, for example if a hardware module cannot supply the necessary functionality on its own. In the following, we refer to all instantiations of underlying functionality, be it hardware or software, as "crypto library".

#### 4.1.1 Supported Algorithms

The following cryptographic algorithms or primitives should be supported by the Crypto Stack:

- Random Number Generation
  - Deterministic Random Number Generator (DRNG)
  - True Random Number Generator (TRNG)
- Symmetric Encryption
  - AES

- \* Key Length: 128 and 256 bits
- \* Modes: ECB, CBC, CTR, GCM, OFB, CFB, XTS
- PRESENT
  - \* Key Length: 128 bits
  - \* Modes: ECB, CBC, CTR, GCM, OFB, CFB, XTS
- ChaCha12/ChaCha20
  - \* Key Length: 256 bits
- AEAD
  - AES GCM
  - ChaCha20 and POLY1305
- Asymmetric Encryption/Decryption and Signature Handling
  - RSA
    - \* Key Length: 1024, 2048, 3072, 4096
    - \* Padding: PKCS#1 v2.2
  - ECDSA
    - \* Key Length: 256 bits
    - \* Curve: NIST P-256
  - Ed25519
- Hash
  - SHA-2
    - \* Length: 224, 256, 384, 512
  - SHA-3
    - \* Length: 224, 256, 384, 512
  - BLAKE
    - \* Length: 224, 256, 384, 512
  - RIPEMD-160
- MAC
  - CMAC
  - GMAC
  - HMAC

- POLY1305
- Key exchange:
  - ECDH
    - \* Key Length: 256 bits
    - \* Curve: NIST P-256
  - X25519

## 4.2 Functional Requirements

### 4.2.1 Crypto Stack

#### 4.2.1.1 General

#### [SRS\_CryptoStack\_00100] Synchronous Job Processing [

<b>Description:</b>	Some crypto services shall allow synchronous job processing.
<b>Rationale:</b>	There are some crypto services which can be calculated very fast and are required very fast. Then, the overhead of the asynchronous job processing including main function calls and call back functions, is too big.
<b>Use Case:</b>	MAC generation for the SecOC module
<b>Dependencies:</b>	–
<b>Supporting Material:</b>	–

]

#### [SRS\_CryptoStack\_00101] Asynchronous Job Processing [

<b>Description:</b>	Some crypto services shall allow asynchronous job processing.
<b>Rationale:</b>	There are some crypto services which require a lot of time or are executed in an HSM. Then, synchronous job processing would require too much time.
<b>Use Case:</b>	Signature verification
<b>Dependencies:</b>	–
<b>Supporting Material:</b>	–

]

**[SRS\_CryptoStack\_00003] The crypto stack shall be able to incorporate modules of the crypto library** [

<b>Description:</b>	The crypto stack shall be able to incorporate modules of a crypto library.
<b>Rationale:</b>	The crypto library itself has to be available in the AUTOSAR stack.
<b>Use Case:</b>	SW implementation of cryptographic primitives.
<b>Dependencies:</b>	–
<b>Supporting Material:</b>	–

]

#### 4.2.1.2 Configuration

**[SRS\_CryptoStack\_00007] The Crypto Stack shall provide scalability for the cryptographic features** [

<b>Description:</b>	The Crypto Stack shall guarantee that the unused cryptographic features are not compiled into the binary.
<b>Rationale:</b>	Different security features require different encryption solutions (example: symmetric/asymmetric encryption, hashing) with or without hardware support. The hardware profiles available offer different features (example: internal NVM, random number generator, secure CPU core...). Scalability of cryptographic features allow different strategies for implementation if some features are not required and thus minimize SW or HW resource utilization.
<b>Use Case:</b>	The mapping between crypto stack and the functionalities of microcontroller hardware allows hardware vendors to develop generic drivers for their HSMs.
<b>Dependencies:</b>	–
<b>Supporting Material:</b>	–

]

**[SRS\_CryptoStack\_00008] The Crypto Stack shall allow static configuration of keys used for cryptographic jobs** [

<b>Description:</b>	The Crypto Stack shall allow static configuration of symmetric and asymmetric key pairs used for crypto services.
<b>Rationale:</b>	It shall be possible to use keys individually.
<b>Use Case:</b>	Data encryption with a protected key in the HSM.
<b>Dependencies:</b>	–
<b>Supporting Material:</b>	–

]

**[SRS\_CryptoStack\_00105] The Crypto Stack shall only allow unique key identifiers** [

<b>Description:</b>	There is one keyId configured for each cryptographic key.
<b>Rationale:</b>	It shall be possible to treat keys individually.
<b>Use Case:</b>	Usage of cryptographic keys.
<b>Dependencies:</b>	–
<b>Supporting Material:</b>	–

]

**[SRS\_CryptoStack\_00013] The modules of the crypto stack shall support only pre-compile time configuration** [

<b>Description:</b>	The modules of the crypto stack shall support only pre-compile time configuration.
<b>Rationale:</b>	No applicable post-build or link-time parameters
<b>Use Case:</b>	All the configurable parameter values must be decided before compile or build time.
<b>Dependencies:</b>	–
<b>Supporting Material:</b>	–

]

**[SRS\_CryptoStack\_00094] The configuration files of the crypto stack modules shall be readable for human beings** [

<b>Description:</b>	The configuration files of the crypto stack modules shall be readable for human beings: e.g. by integration of comments or by tool - support.
<b>Rationale:</b>	Human being have to read and understand the configuration. So the configuration shall be readable and understandable for human being.
<b>Use Case:</b>	Debugging
<b>Dependencies:</b>	–
<b>Supporting Material:</b>	–

]

#### 4.2.1.3 Initialization

None

#### 4.2.1.4 Normal Operation

**[SRS\_CryptoStack\_00009] The Crypto Stack shall support reentrancy for all crypto services** [

<b>Description:</b>	The Crypto Stack shall support reentrancy of crypto related interfaces to enable parallel operations of the same or different type when requested by multiple users. This requirement also covers scenarios where applications are residing on different cores.
<b>Rationale:</b>	Crypto jobs shall be processable simultaneously
<b>Use Case:</b>	Different applications may use cryptographic services in parallel. Handling of different tasks at the same time is necessary.
<b>Dependencies:</b>	–
<b>Supporting Material:</b>	–

]

**[SRS\_CryptoStack\_00010] The Crypto Stack shall conceal symmetric keys from the users of crypto services** [

<b>Description:</b>	There shall be no interface to extract symmetric key values directly to the user. Keys shall be addressed via identifiers by the users. Such keys shall only be exported in an encrypted format.
<b>Rationale:</b>	If keys are stored in the application, this increases the chances of invalidation of keys or keys being compromised.
<b>Use Case:</b>	Keys residing in the HSM
<b>Dependencies:</b>	–
<b>Supporting Material:</b>	–

]

**[SRS\_CryptoStack\_00011] The Crypto Stack shall conceal asymmetric private keys from the users of Crypto services** [

<b>Description:</b>	There shall be no interface to extract asymmetric private key values directly to the user. Keys shall be addressed via identifiers by the Users. Such keys shall only be exported in an encrypted format.
<b>Rationale:</b>	If keys are stored in the application, this increases the chances of invalidation of keys or keys being compromised.
<b>Use Case:</b>	Keys residing in the HSM
<b>Dependencies:</b>	–
<b>Supporting Material:</b>	–

]

**[SRS\_CryptoStack\_00019] The Crypto Stack shall identify random number generation as a cryptographic primitive which can be requested to a driver [**

<b>Description:</b>	The Crypto Stack shall identify random number generation as a cryptographic primitive which can be requested to a driver.
<b>Rationale:</b>	Random number Generators residing on different crypto drivers should be accessed using a homogenous interface.
<b>Use Case:</b>	Generate random number
<b>Dependencies:</b>	–
<b>Supporting Material:</b>	–

]

**[SRS\_CryptoStack\_00020] The Crypto Stack shall identify symmetric encryption/decryption as a cryptographic primitive which can be requested to a driver [**

<b>Description:</b>	The Crypto Stack shall identify symmetric encryption/decryption as a cryptographic primitive which can be requested to a driver.
<b>Rationale:</b>	Symmetric algorithms residing on different crypto drivers should be accessed using a homogenous interface.
<b>Use Case:</b>	Encrypted communication
<b>Dependencies:</b>	–
<b>Supporting Material:</b>	–

]

**[SRS\_CryptoStack\_00021] The Crypto Stack shall identify asymmetric encryption/decryption as a cryptographic primitive which can be requested to a driver [**

<b>Description:</b>	The Crypto Stack shall identify asymmetric encryption/decryption as a cryptographic primitive which can be requested to a driver.
<b>Rationale:</b>	Asymmetric algorithms residing on different crypto drivers should be accessed using a homogenous interface.
<b>Use Case:</b>	Unique Interface for success of heterogeneous hardware- and software-solutions
<b>Dependencies:</b>	–
<b>Supporting Material:</b>	–

]

**[SRS\_CryptoStack\_00022] The Crypto Stack shall identify MAC generation/verification as a cryptographic primitive which can be requested to a driver [**

<b>Description:</b>	The Crypto Stack shall identify MAC generation/verification as a cryptographic primitive which can be requested to a driver.
<b>Rationale:</b>	MAC algorithms residing on different crypto drivers should be accessed using a homogenous interface.
<b>Use Case:</b>	SecOC using MACs to verify messages
<b>Dependencies:</b>	–
<b>Supporting Material:</b>	–

]

**[SRS\_CryptoStack\_00023] The Crypto Stack shall identify asymmetric signature generation/verification as a cryptographic primitive which can be requested to a driver [**

<b>Description:</b>	The Crypto Stack shall identify asymmetric signature generation/verification as a cryptographic primitive which can be requested to a driver.
<b>Rationale:</b>	Asymmetric signature algorithms residing on different crypto drivers should be accessed using a homogenous interface.
<b>Use Case:</b>	Signature creation/verification
<b>Dependencies:</b>	–
<b>Supporting Material:</b>	–

]

**[SRS\_CryptoStack\_00024] The Crypto Stack shall identify hash calculation as a cryptographic primitive which can be requested to a driver [**

<b>Description:</b>	The Crypto Stack shall identify hash calculation as a cryptographic primitive which can be requested to a driver.
<b>Rationale:</b>	Hash algorithms residing on different crypto drivers should be accessed using a homogenous interface.
<b>Use Case:</b>	Signature verification
<b>Dependencies:</b>	–
<b>Supporting Material:</b>	–

]

**[SRS\_CryptoStack\_00026] The Crypto Stack shall provide an interface for the generation of asymmetric keys [**

<b>Description:</b>	The Crypto Stack shall provide an abstracted interface for the generation of asymmetric key pair service.
<b>Rationale:</b>	Key generation services residing on different Crypto drivers should be accessed using a homogenous interface.
<b>Use Case:</b>	Generation of an asymmetric key pair inside the ECU. Then, the private key never has to be available outside the ECU.
<b>Dependencies:</b>	–
<b>Supporting Material:</b>	–

]

**[SRS\_CryptoStack\_00027] The Crypto Stack shall provide an interface for the generation of symmetric keys [**

<b>Description:</b>	The Crypto Stack shall abstract the user from multiple symmetric keys stored by various Crypto Drivers through a standardized interface. Also, it shall provide an interface to the driver for generation of such keys.
<b>Rationale:</b>	Key generation services residing on different Crypto drivers should be accessed using a homogenous interface.
<b>Use Case:</b>	Password-based key input
<b>Dependencies:</b>	–
<b>Supporting Material:</b>	–

]

**[SRS\_CryptoStack\_00103] The Crypto Stack shall provide an interface for the derivation of symmetric keys [**

<b>Description:</b>	The Crypto Stack shall abstract the user from multiple symmetric keys stored by various Crypto Drivers through a standardized interface. Also, it shall provide an interface to the driver for derivation of such keys.
<b>Rationale:</b>	Key derivation services residing on different Crypto drivers should be accessed using a homogenous interface.
<b>Use Case:</b>	Password-based key input
<b>Dependencies:</b>	–
<b>Supporting Material:</b>	–

]

**[SRS\_CryptoStack\_00028] The Crypto Stack shall provide an interface for key exchange mechanisms** [

<b>Description:</b>	The Crypto Stack shall support key exchange mechanism as a key management interface
<b>Rationale:</b>	Key exchange algorithms residing on different crypto drivers should be accessed using a homogenous interface
<b>Use Case:</b>	Session handling
<b>Dependencies:</b>	–
<b>Supporting Material:</b>	–

]

**[SRS\_CryptoStack\_00029] The Crypto Stack shall provide an interface for key wrapping/extraction mechanisms** [

<b>Description:</b>	The Crypto Stack shall support key wrapping (encapsulation) and extraction mechanism forward such requests from CSM to the respective driver. It shall support wrapping using a symmetric key as well as asymmetric key.
<b>Rationale:</b>	Key wrapping and encapsulation algorithms implemented in the driver shall not be accessed directly by the users and need to be abstracted.
<b>Use Case:</b>	Session handling
<b>Dependencies:</b>	–
<b>Supporting Material:</b>	–

]

**[SRS\_CryptoStack\_00031] The Crypto Stack shall provide an interface for parsing certificates** [

<b>Description:</b>	The Crypto Stack shall support parsing certificates and extracting the contained keys
<b>Rationale:</b>	The crypto driver shall parse incoming certificates and store the key information in the corresponding key
<b>Use Case:</b>	For PKI it is necessary to obtain public keys out of certificates
<b>Dependencies:</b>	–
<b>Supporting Material:</b>	–

]

**[SRS\_CryptoStack\_00061] The Crypto Stack shall support detection of invalid keys** [

<b>Description:</b>	The implementation of a cryptographic primitive shall detect and reject invalid keys.
<b>Rationale:</b>	Algorithms like RSA or several ECC flavors know keys which can be used to perform the mathematical foundation of the algorithm without an error but address special corner cases and are not secure to handle. Keys like that have to be identified and rejected. There is no generic approach hence the implementation has to be in the cryptographic primitive itself or, in case hardware is used, in its driver.
<b>Use Case:</b>	RSA and several Elliptic Curve Cryptosystems
<b>Dependencies:</b>	–
<b>Supporting Material:</b>	–

]

#### 4.2.1.5 Shutdown Operation

None

#### 4.2.1.6 Fault Operation

None

### 4.2.2 Key Manager

#### 4.2.2.1 General

**[SRS\_CryptoStack\_00106] Key manager operation shall either run synchronously or asynchronously.** [

<b>Description:</b>	A key manager operation shall support synchronous mode or asynchronous mode, if the computation takes a long time. . For the latter case, a callback shall indicate that an operation has been completed.
<b>Rationale:</b>	To avoid multiple task activation a key management operation shall be executed in the background.
<b>Use Case:</b>	Key derivation can take considerable amount of time
<b>Dependencies:</b>	–
<b>Supporting Material:</b>	Concept 636 "Security Extensions" - C2

]

**[SRS\_CryptoStack\_00107] Key manager shall provide interfaces to generate or update key material. [**

<b>Description:</b>	The KeyM module shall provide an API allowing a software component or DCM to generate or update the key material for an ECU. KeyM shall use the CSM for the required cryptographic operations and key storage
<b>Rationale:</b>	A basic software module e.g. DCM or software component) triggers to generate or re-generate key material. The KeyM generates the key material and stores it in the crypto driver by calling the appropriate functions of the CSM.
<b>Use Case:</b>	Key server wants to provide key material
<b>Dependencies:</b>	–
<b>Supporting Material:</b>	Concept 636 "Security Extensions" - C2

]

**[SRS\_CryptoStack\_00108] Key manager shall be able to negotiate a shared secret by exchanging messages with other ECUs [**

<b>Description:</b>	The KeyM module shall support key negotiation between ECUs via interfaces to the PduR.
<b>Rationale:</b>	Key management tasks such as multi-party key negotiation can require the exchange of messages to agree on keys.
<b>Use Case:</b>	Onboard negotiation of keys
<b>Dependencies:</b>	–
<b>Supporting Material:</b>	Concept 636 "Security Extensions" - C2

]

**[SRS\_CryptoStack\_00109] Key manager shall be able to manage derivation of key material from a common secret [**

<b>Description:</b>	The KeyM module shall provide an API for deriving keys from a common shared secret.
<b>Rationale:</b>	Secret keys (e.g. for secure onboard communication) have to be shared between multiple ECUs. One way to handle this is to derive the keys from a secret which is shared between the participating ECUs.
<b>Use Case:</b>	The key server provides a common secret and the key manager derives several symmetric keys out of this common secret.
<b>Dependencies:</b>	–
<b>Supporting Material:</b>	Concept 636 "Security Extensions" - C2

]

**[SRS\_CryptoStack\_00110] The KeyM module shall support on-board generated keys** [

<b>Description:</b>	The KeyM shall provide an API for handling of on-board generated keys.
<b>Rationale:</b>	On-board generated keys have to be negotiated between two or more ECUs.
<b>Use Case:</b>	C2 - Key Management
<b>Dependencies:</b>	–
<b>Supporting Material:</b>	Concept 636 "Security Extensions" - C2

]

**[SRS\_CryptoStack\_00111] The KeyM module shall support verification of certificates based on configured rules** [

<b>Description:</b>	The certificate verification operations provided by KeyM shall allow to check the values of configured certificate elements as part of the verification operation.
<b>Rationale:</b>	Besides the verification of signatures and basic certificate elements, certificate verification may require checks whether custom certificate elements conform to specific rules.
<b>Use Case:</b>	C2 - Key Management
<b>Dependencies:</b>	–
<b>Supporting Material:</b>	Concept 636 "Security Extensions" - C2

]

**[SRS\_CryptoStack\_00112] The KeyM module shall support retrieving arbitrary elements of a certificate** [

<b>Description:</b>	KeyM shall provide operations to retrieve the value of arbitrary certificate elements
<b>Rationale:</b>	Applications and basic software modules may need to use the value of specific certificate elements to perform operations.
<b>Use Case:</b>	C2 - Key Management
<b>Dependencies:</b>	–
<b>Supporting Material:</b>	Concept 636 "Security Extensions" - C2

]

**[SRS\_CryptoStack\_00125] AUTOSAR certificate handler shall support the certificate formats CVC and X.509.** [

<b>Description:</b>	AUTOSAR certificate handler shall support the certificate formats CVC and X.509.
<b>Rationale:</b>	Only a subset of common known and used client certificate types is supported by AUTOSAR. This allows a standardized handling and evaluation of certificates and content.

▽



<b>Use Case:</b>	Certificates issued by OEM are supported by the ECU. E.g. The OEM PKI issues a certificate for a repair shop diagnostic tester. The diagnostic tester authenticates itself with the ECU.
<b>Dependencies:</b>	–
<b>Supporting Material:</b>	Concept 636 "Security Extensions" - C2

]

#### 4.2.2.2 Configuration

##### [SRS\_CryptoStack\_00113] Keys in the crypto stack can be uniquely identified [

<b>Description:</b>	All keys maintained by KeyM must be uniquely addressed so that new key material provided by the key server can be assigned to the corresponding key in the crypto driver.
<b>Rationale:</b>	A symbolic identification is required so that KeyM can uniquely identify which key is used by a corresponding job.
<b>Use Case:</b>	C2 - Key Management
<b>Dependencies:</b>	–
<b>Supporting Material:</b>	Concept 636 "Security Extensions" - C2

]

##### [SRS\_CryptoStack\_00114] Crypto driver shall place keys into specific key slots [

<b>Description:</b>	A key shall be placed to a specific key slot. This is required for the SHE to provide the correct key update information.
<b>Rationale:</b>	The SHE update protocol information includes the slot of the key that shall be updated.
<b>Use Case:</b>	C2 - Key Management
<b>Dependencies:</b>	–
<b>Supporting Material:</b>	Concept 636 "Security Extensions" - C2

]

##### [SRS\_CryptoStack\_00115] KeyM shall be highly configurable to support different OEM use cases [

<b>Description:</b>	The key management provides various options like certificate handling, key management operation. Also, there is already a various number of key management systems established that must be fulfilled with this module. This requires a high flexibility of the module to adapt it to different use cases.
---------------------	--



△

<b>Rationale:</b>	The Key Management depends highly on OEM specific processes and formats.
<b>Use Case:</b>	C2 - Key Management
<b>Dependencies:</b>	–
<b>Supporting Material:</b>	Concept 636 "Security Extensions" - C2

]

#### 4.2.2.3 Initialization

##### [SRS\_CryptoStack\_00116] Keys shall use a default value if configured [

<b>Description:</b>	It shall be possible to configure KeyM to use key default values.
<b>Rationale:</b>	A key server may not be available during development of a secure system. So it should be possible to configure default keys which can be used during development.
<b>Use Case:</b>	System tests during development phase.
<b>Dependencies:</b>	–
<b>Supporting Material:</b>	Concept 636 "Security Extensions" - C2

]

##### [SRS\_CryptoStack\_00117] Keys shall not be used if they are empty or corrupted [

<b>Description:</b>	A key that has no default value or has been programmed but its contents cannot be retrieved shall be marked as invalid and is therefore not usable.
<b>Rationale:</b>	A system that has no properly initialized keys shall not provide any undetermined cryptographic results.
<b>Use Case:</b>	C2 - Key Management
<b>Dependencies:</b>	–
<b>Supporting Material:</b>	Concept 636 "Security Extensions" - C2

]

#### 4.2.2.4 Normal Operation

**[SRS\_CryptoStack\_00118] Key material shall be securely stored either in NVM or CSM** [

<b>Description:</b>	The KeyM shall use CSM interfaces for securely storing keys.
<b>Rationale:</b>	Off-board generated keys and on-board generated keys have to be securely stored.
<b>Use Case:</b>	C2 - Key Management
<b>Dependencies:</b>	–
<b>Supporting Material:</b>	Concept 636 "Security Extensions" - C2

]

**[SRS\_CryptoStack\_00119] Provide a proof that the key has been programmed correctly** [

<b>Description:</b>	The KeyM shall provide interfaces for providing a proof for correctly programmed keys. A verification function can be used to provide a proof to the key server if the correct key is programmed and associated with a specific job.
<b>Rationale:</b>	Verification of a succesful key installation. A key provider may want to check if the key has been programmed correctly.
<b>Use Case:</b>	C2 - Key Management
<b>Dependencies:</b>	–
<b>Supporting Material:</b>	Concept 636 "Security Extensions" - C2

]

#### 4.2.2.5 Shutdown Operation

**[SRS\_CryptoStack\_00120] Cleanup all key material on shutdown operation** [

<b>Description:</b>	Keys may be temporarily stored in RAM. A shutdown operation may cleanup these keys by removing the key data from RAM.
<b>Rationale:</b>	Keys may be stored during shutdown operation. For security reason, keys in RAM shall be destroyed.
<b>Use Case:</b>	C2 - Key Management
<b>Dependencies:</b>	–
<b>Supporting Material:</b>	Concept 636 "Security Extensions" - C2

]

#### 4.2.2.6 Fault Operation

### [SRS\_CryptoStack\_00121] Pass security related events to security event memory (SEM) for secure logging

Status: DRAFT

[

<b>Description:</b>	The KeyM shall use the interfaces provided by the security event memory for secure logging of security related events which are produced by KeyM.
<b>Rationale:</b>	Events such as failed key negotiation are security related events which need to be logged.
<b>Use Case:</b>	C2 - Key Management
<b>Dependencies:</b>	–
<b>Supporting Material:</b>	Concept 636 "Security Extensions" - C2

]

#### 4.2.3 Crypto Service Manager

##### 4.2.3.1 General

### [SRS\_CryptoStack\_00006] Each primitive of the CRYIF shall belong to exactly one service of the CSM [

<b>Description:</b>	Each primitive of the CRYIF shall belong to exactly one service of the CSM.
<b>Rationale:</b>	There are channels which map a user specific crypto primitive via the CRYIF to the underlying Crypto Driver module.
<b>Use Case:</b>	The CRYIF is responsible for each service to map to the corresponding Crypto Driver module
<b>Dependencies:</b>	–
<b>Supporting Material:</b>	–

]

#### 4.2.3.2 Configuration

**[SRS\_CryptoStack\_00079] The job processing mode (synchronous or asynchronous) of a CSM service shall be defined by static configuration [**

<b>Description:</b>	The mode of cryptographic jobs provided by the CSM shall be defined by static configuration.
<b>Rationale:</b>	It shall not be possible to change the behavior of a specific CSM service during runtime.
<b>Use Case:</b>	Synchronous hash calculation
<b>Dependencies:</b>	–
<b>Supporting Material:</b>	–

]

**[SRS\_CryptoStack\_00102] The priority of a user and its crypto jobs shall be defined by static configuration [**

<b>Description:</b>	The user's priority shall be defined by static configuration. All jobs of that user inherit that priority.
<b>Rationale:</b>	There are crypto jobs which have to be processed very fast (e.g. MAC generation for the SecOC). Other crypto jobs (e.g. hashing over the whole ROM) take very long but are not time critical.
<b>Use Case:</b>	Prioritized job processing
<b>Dependencies:</b>	–
<b>Supporting Material:</b>	–

]

**[SRS\_CryptoStack\_00080] The set of cryptographic services provided by the CSM shall be defined by static configuration [**

<b>Description:</b>	The set of cryptographic services provided by the CSM shall be defined by static configuration.
<b>Rationale:</b>	It is not possible during runtime to add new CSM services.
<b>Use Case:</b>	If symmetrical encryption is supported by the driver, it has to be configured which user with which key is using it.
<b>Dependencies:</b>	–
<b>Supporting Material:</b>	–

]

**[SRS\_CryptoStack\_00081] The CSM module specification shall specify which other modules are required** [

<b>Description:</b>	The CSM module specification shall specify which other modules are required.
<b>Rationale:</b>	Basic functionality
<b>Use Case:</b>	–
<b>Dependencies:</b>	–
<b>Supporting Material:</b>	–

]

**[SRS\_CryptoStack\_00082] The CSM module specification shall specify the interface and behavior of the callback function, if the asynchronous job processing mode is selected** [

<b>Description:</b>	The CSM module specification shall specify how the callback function has to be implemented, if the asynchronous job processing mode is selected.
<b>Rationale:</b>	The CSM has to call the callback function. Thus, the CSM has to know the signature of the callback function.
<b>Use Case:</b>	–
<b>Dependencies:</b>	–
<b>Supporting Material:</b>	–

]

#### 4.2.3.3 Initialization

None

#### 4.2.3.4 Normal Operation

**[SRS\_CryptoStack\_00084] The CSM module shall use the streaming approach for some selected services** [

<b>Description:</b>	The CSM module shall use the streaming approach for some provided services (see Software Specification of CSM).
<b>Rationale:</b>	Basic functionality
<b>Use Case:</b>	It shall be possible to hand over the input data in small chunks to the service.
<b>Dependencies:</b>	–

]

#### 4.2.3.5 Shutdown Operation

<what to do when the module is shut down>

#### 4.2.3.6 Fault Operation

**[SRS\_CryptoStack\_00086] The CSM module shall distinguish between error types [**

<b>Description:</b>	The CSM module shall distinguish between the following two types or errors: - errors that can only occur during development - errors that are expected to occur also in production code
<b>Rationale:</b>	Basic functionality
<b>Use Case:</b>	–
<b>Dependencies:</b>	–
<b>Supporting Material:</b>	–

]

**[SRS\_CryptoStack\_00087] The CSM module shall report detected development errors to the Default Error Tracer [**

<b>Description:</b>	The CSM module shall report detected development errors to the Default Error Tracer
<b>Rationale:</b>	Basic functionality
<b>Use Case:</b>	–
<b>Dependencies:</b>	–
<b>Supporting Material:</b>	–

]

**[SRS\_CryptoStack\_00096] The CSM module shall not return specific development error codes via the API [**

<b>Description:</b>	The CSM module shall not return specific development error codes via the API. In case of a detected development error the error shall only be reported to the DET. If the API function which detected the error has the return type Std_ReturnType, it shall return E_NOT_OK.
<b>Rationale:</b>	Basic functionality
<b>Use Case:</b>	–
<b>Dependencies:</b>	–
<b>Supporting Material:</b>	–

]

**[SRS\_CryptoStack\_00097] The CSM shall check passed API parameters for validity** [

<b>Description:</b>	The CSM shall check passed API parameters for validity. This checking shall be statically configurable for those errors that only can occur during development.
<b>Rationale:</b>	Basic functionality
<b>Use Case:</b>	Debugging
<b>Dependencies:</b>	–
<b>Supporting Material:</b>	–

]

## 4.2.4 Crypto Interface

### 4.2.4.1 Configuration

**[SRS\_CryptoStack\_00014] The Crypto Interface shall have an interface to the static configuration information of the Crypto Driver** [

<b>Description:</b>	The Crypto Interface shall have an interface to the static configuration information of the Crypto Driver.
<b>Rationale:</b>	Flexibility and scalability
<b>Use Case:</b>	To derive vendor API Infix to support multiple Crypto Drivers
<b>Dependencies:</b>	–
<b>Supporting Material:</b>	–

]

**[SRS\_CryptoStack\_00015] Channels mapped to different Crypto Driver Objects shall be uniquely configurable in Crypto Interface** [

<b>Description:</b>	The Crypto Interface shall support a configuration model where all virtual channels shall be statically mapped to Crypto Driver Objects. Virtual channels are the virtual way from the queue of the CSM over the CRYIF to the corresponding Crypto Driver Object.
<b>Rationale:</b>	Each crypto driver object in the driver can be abstracted from and utilized by CSM using virtual channels.
<b>Use Case:</b>	Two hardware resources: one for symmetric cryptography, one for asymmetric cryptography
<b>Dependencies:</b>	–
<b>Supporting Material:</b>	–

]

#### 4.2.4.2 Initialization

None

#### 4.2.4.3 Normal Operation

None

#### 4.2.4.4 Shutdown Operation

None

#### 4.2.4.5 Fault Operation

**[SRS\_CryptoStack\_00034] The Crypto Interface shall report detected development errors to the Default Error Tracer** [

<b>Description:</b>	The Crypto Interface shall report detected development errors to the Default Error Tracer (DET). The detection and reporting shall be statically configurable with one single preprocessor switch.
<b>Rationale:</b>	Debugging Support
<b>Use Case:</b>	All the input parameters and internal states have to be validated before processing.
<b>Dependencies:</b>	–
<b>Supporting Material:</b>	–

]

### 4.2.5 Crypto Driver

#### 4.2.5.1 Configuration

**[SRS\_CryptoStack\_00036] The Crypto Driver shall allow static configuration of Crypto Driver Objects** [

<b>Description:</b>	The Crypto Driver shall allow defining of different Crypto Driver Objects.
<b>Rationale:</b>	Abstraction of different hardware units
<b>Use Case:</b>	Parallel processing of symmetric operations and asymmetric operations
<b>Dependencies:</b>	–

▽

△

<b>Supporting Material:</b>	–
-----------------------------	---

]

**[SRS\_CryptoStack\_00104] Crypto Interface keys mapped to different Crypto Driver Keys shall be uniquely configurable in the Crypto Interface [**

<b>Description:</b>	The Crypto Interface shall support a configuration model where all CRYIF keys shall be statically mapped to keys in the crypto driver.
<b>Rationale:</b>	Similar to the channels where the CsmQueue is mapped to the Crypto Driver Object, the keys in the CSM have to mapped via the CRYIF to the corresponding key in the Crypto Driver.
<b>Use Case:</b>	Multiple Crypto Driver modules where each module has its own key identifiers
<b>Dependencies:</b>	–
<b>Supporting Material:</b>	–

]

**4.2.5.2 Initialization**

None

**4.2.5.3 Normal Operation**

**[SRS\_CryptoStack\_00098] The Crypto Driver shall provide access to all cryptographic algorithms supported by the hardware [**

<b>Description:</b>	The Crypto Driver shall support access to all by the Crypto Stack supported algorithms.
<b>Rationale:</b>	Usage of hardware support and performance benefits.
<b>Use Case:</b>	Primitives which are supported by the HSM should be accessible through the Crypto Driver
<b>Dependencies:</b>	–
<b>Supporting Material:</b>	–

]

**4.2.5.4 Shutdown Operation**

None

#### 4.2.5.5 Fault Operation

None

### 4.2.6 Security Event Memory

#### 4.2.6.1 General

#### [SRS\_CryptoStack\_00122] Log security events reported by basic software modules and SWC [

<b>Description:</b>	The security event memory (SEM) shall enable to log security events (SE) which are monitored by BSW modules or SWCs. It shall be possible to store the type of the security event along with data which is useful for forensic analysis.
<b>Rationale:</b>	Log security events in a way that is useful for forensic analysis
<b>Use Case:</b>	UC1.1: BSW modules or SWCs write security related events to the SEM
<b>Dependencies:</b>	–
<b>Supporting Material:</b>	Concept 636 "Security Extensions" - C1

]

#### [SRS\_CryptoStack\_00123] Configure security event properties [

<b>Description:</b>	Depending on the type of security event the number of storable snap shots and the properties of the snap shots shall be configurable.
<b>Rationale:</b>	Depending on the type of event different data needs to be stored in the snapshot for the event. Depending on the expected frequency of the event specific numbers of storable snapshots are sufficient.
<b>Use Case:</b>	UC1.1: BSW modules or SWCs write security related events to the SEM
<b>Dependencies:</b>	–
<b>Supporting Material:</b>	Concept 636 "Security Extensions" - C1

]

#### [SRS\_CryptoStack\_00124] Allow authorized users to read SEM data via diagnostic interfaces [

<b>Description:</b>	The SEM shall provide SEM data via diagnostic interfaces. SEM data shall be protected from unauthorized read and write access via diagnostic interfaces.
<b>Rationale:</b>	Diagnostic testers are used to access SEM data
<b>Use Case:</b>	UC1.2: Authorized stakeholders read data from the SEM for off-board forensic analysis
<b>Dependencies:</b>	–

▽

△

<b>Supporting Material:</b>	Concept 636 "Security Extensions" - C1
-----------------------------	--

]

## 4.3 Non-Functional Requirements (Qualities)

### 4.3.1 General

None

### 4.3.2 Crypto Service Manager

**[SRS\_CryptoStack\_00088] The CSM module shall provide an abstraction layer which offers a standardized interface to higher software layers to access cryptographic algorithms [**

<b>Description:</b>	The CSM module shall provide an abstraction layer which offers a standardized interface to higher software layers to access cryptographic algorithms.
<b>Rationale:</b>	An abstraction layer encapsulates internal behaviors and reduces complexity. It also increases maintainability, improves portability and eases testability.
<b>Use Case:</b>	Session handling.
<b>Dependencies:</b>	–
<b>Supporting Material:</b>	–

]

**[SRS\_CryptoStack\_00089] The CSM module shall be located in the AUTOSAR service layer [**

<b>Description:</b>	The CSM module shall be located in the Autosar service layer
<b>Rationale:</b>	Management functionality must be available to all modules and layers of the system
<b>Use Case:</b>	The CSM module shall be accessible from applications above the RTE.
<b>Dependencies:</b>	–
<b>Supporting Material:</b>	–

]

**[SRS\_CryptoStack\_00090] The CSM shall provide an interface to be accessible via the RTE** [

<b>Description:</b>	The CSM shall provide an interface to be accessible via the RTE.
<b>Rationale:</b>	The CSM module shall be accessible from applications above the RTE.
<b>Use Case:</b>	Applications which require crypto services.
<b>Dependencies:</b>	–
<b>Supporting Material:</b>	–

]

**[SRS\_CryptoStack\_00091] The CSM shall provide one Provide-Port for each configuration** [

<b>Description:</b>	The CSM shall provide one Provide-Port for each configuration. All configured services shall be accessible via this port.
<b>Rationale:</b>	All crypto services shall be accessible from applications above the RTE via the Provide-Port.
<b>Use Case:</b>	All applications request to the CSM uses this port.
<b>Dependencies:</b>	–
<b>Supporting Material:</b>	–

]

**[SRS\_CryptoStack\_00092] The CSM shall provide one Require-Port for each configuration** [

<b>Description:</b>	The CSM shall provide one Require-Port for each configuration. The configured callback function shall be accessible via this port.
<b>Rationale:</b>	All crypto services shall have access to the configured callback functions via the Require-Port.
<b>Use Case:</b>	Most asynchronous services own a callback function.
<b>Dependencies:</b>	–
<b>Supporting Material:</b>	–

]

### 4.3.3 Crypto Interface

**[SRS\_CryptoStack\_00075] The Crypto Interface shall be the interface layer between the underlying crypto driver(s) and upper layers** [

<b>Description:</b>	The Crypto Interface is the single interface for all upper Layer (BSW) for crypto operations. The Crypto Interface is the single user of the Crypto Driver
<b>Rationale:</b>	Interfaces and interaction
<b>Use Case:</b>	Different Users might need to access more than one Crypto Drivers or SW based solutions. Also, different upper layers might need to access crypto services.
<b>Dependencies:</b>	–
<b>Supporting Material:</b>	AUTOSAR_WP Architecture_SoftwareArchitecture

]

**[SRS\_CryptoStack\_00076] The Crypto Interface implementation and interface shall be independent from underlying Crypto Hardware or Software** [

<b>Description:</b>	The CRYIF implementation and CRYIF interfaces shall be independent from the underlying Crypto Driver modules.
<b>Rationale:</b>	Portability and reusability
<b>Use Case:</b>	Encapsulate implementation details of a specific Crypto module from higher software layers.
<b>Dependencies:</b>	–
<b>Supporting Material:</b>	–

]

### 4.3.4 Crypto Driver

**[SRS\_CryptoStack\_00095] The Crypto Driver module shall strictly separate error and status information** [

<b>Description:</b>	The Crypto Driver module shall strictly separate error and status information. This requirement applies to return values and also to internal variables.
<b>Rationale:</b>	The distinction between error and status information allow an easier handling with them. Errors shall be treated differently than status information.
<b>Use Case:</b>	All return values are error information.
<b>Dependencies:</b>	–
<b>Supporting Material:</b>	–

]

## 5 References

- [1] Specification of Crypto Service Manager  
AUTOSAR\_CP\_SWS\_CryptoServiceManager
- [2] Specification of Crypto Interface  
AUTOSAR\_CP\_SWS\_CryptoInterface
- [3] Specification of Crypto Driver  
AUTOSAR\_CP\_SWS\_CryptoDriver
- [4] Specification of Key Manager  
AUTOSAR\_CP\_SWS\_KeyManager
- [5] Standardization Template  
AUTOSAR\_FO\_TPS\_StandardizationTemplate
- [6] Glossary  
AUTOSAR\_FO\_TR\_Glossary

## **A Change history of AUTOSAR traceable items**

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

### **A.1 Traceable item history of this document according to AUTOSAR Release R25-11**

#### **A.1.1 Added Requirements in R25-11**

none

#### **A.1.2 Changed Requirements in R25-11**

none

#### **A.1.3 Deleted Requirements in R25-11**

none