Document Title	Requirements on CryptoStack Stack
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	426
Document Classification	Auxiliary

Document Status	Final
Part of AUTOSAR Standard	Classic Platform
Part of Standard Release	4.3.0

Document Change History

Date	Release	Changed by	Change Description
2016-11-30	4.3.0	AUTOSAR Release Management	• Added requirements for the whole Crypto Stack and renamed the document • Introduced crypto job concept • Introduced key management concept
2014-10-31	4.2.1	AUTOSAR Release Management	• Editorial changes
2013-10-31	4.1.2	AUTOSAR Release Management	• Editorial changes
2013-03-15	4.1.1	AUTOSAR Administration	• TPS_STDT_0078 formatting • Traceability of BSWAndRTE_Features
2010-09-30	3.1.5	AUTOSAR Administration	• Initial release

Disclaimer

Advice for users

AUTOSAR specifications may contain exemplary items (exemplary reference models, "use cases", and/or references to exemplary technical solutions, devices, processes or software).

Any such exemplary items are contained in the specifications for illustration purposes only, and they themselves are not part of the AUTOSAR Standard. Neither their presence in such specifications, nor any later documentation of AUTOSAR conformance of products actually implementing such exemplary items, imply that intellectual property rights covering such exemplary items are licensed under the same rules as applicable to the AUTOSAR Standard.

Table of Contents

1 Scope of Document

This document specifies the requirements of the crypto stack:

- Crypto Service Manager (Csm),
- Crypto Interface (CryIf) and
- Crypto Driver (Crypto).

2 Conventions to be used

- The representation of requirements in AUTOSAR documents follows the table specified in [TPS_STDT_00078].

- In requirements, the following specific semantics shall be used (based on the Internet Engineering Task Force IETF).

  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:

  - SHALL: This word 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.
  - MUST: This word means that the definition is an absolute requirement of the specification due to legal issues.
  - MUST NOT: This phrase means that the definition is an absolute prohibition of the specification due to legal constraints.
  - SHOULD: This word, or the adjective "RECOMMENDED", mean 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" mean 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, MUST 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, MUST 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

Abbreviation / Acronym:	Description
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
CTR	Counter
DEM / Dem	Diagnostic Event 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
ECIES	Elliptic Curve Integrated Encryption Scheme
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
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
SHE	Secure Hardware Extension
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

3.1 Glossary of Terms

Terms:	***Description:***	
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.	
Crypto Primitive	A crypto primitive is an instance of a configured cryptographic algorithm.	
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.	
Secure Counter	"Secure" in the context of Secure Counter means, that the counter shall be secured against direct access of the user. Thus, for a meaningful implementation, the Secure Counter should be resident in a special secured nonvolatile ROM (e.g. in an HSM)	

4 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 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
- Asymmetric Encryption/Decryption and Signature Handling
  - RSA
    - Key Length: 1024, 2048, 3072, 4096
    - Padding: PKCS#1 v2.2
  - Curve25519/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

5 Requirements Specification

5.1 Functional Requirements

5.1.1 Crypto Stack

5.1.1.1 General

5.1.1.1.1 [SRS_CryptoStack_00100] Synchronous Job Processing

[

Type:	Valid
Description:	Some crypto services shall allow synchronous job processing.
Rationale:	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.
Use Case:	MAC generation for the SecOC module
Dependencies:	--
Supporting Material:	--

](RS_BRF_01456)

5.1.1.1.2 [SRS_CryptoStack_00101] Asynchronous Job Processing

[

Type:	Valid
Description:	Some crypto services shall allow asynchronous job processing.
Rationale:	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.
Use Case:	Signature verification
Dependencies:	--
Supporting Material:	--

](RS_BRF_01456)

5.1.1.1.3 [SRS_CryptoStack_00003] The crypto stack shall be able to incorporate modules of the crypto library

[

Type:	Valid
Description:	The crypto stack shall be able to incorporate modules of a crypto library.
Rationale:	The crypto library itself has to be available in the AUTOSAR stack.
Use Case:	SW implementation of cryptographic primitives.
Dependencies:	--
Supporting Material:	--

](RS_BRF_02032)

5.1.1.2 Configuration

5.1.1.2.1 [SRS_CryptoStack_00007] The Crypto Stack shall provide scalability for the cryptographic features

[

Type:	Valid

Description:	The Crypto Stack shall guarantee that the unused cryptographic features are not compiled into the binary.
Rationale:	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, counters, 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.
Use Case:	The mapping between crypto stack and the functionalities of microcontroller hardware allows hardware vendors to develop generic drivers for their HSMs.
Dependencies:	--
Supporting Material:	--

](RS_BRF_01456, RS_BRF_02031)

5.1.1.2.2 [SRS_CryptoStack_00008] The Crypto Stack shall allow static configuration of keys used for cryptographic jobs

[

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

](RS_BRF_02031, RS_BRF_01946)

5.1.1.2.3 [SRS_CryptoStack_00105] The Crypto Stack shall only allow unique key identifiers

[

Type:	Valid
Description:	There is one keyId configured for each cryptographic key.
Rationale:	It shall be possible to treat keys individually.
Use Case:	Usage of cryptographic keys.
Dependencies:	--
Supporting Material:	--

](RS_BRF_02031, RS_BRF_01946)

5.1.1.2.4 [SRS_CryptoStack_00013] The modules of the crypto stack shall support only pre-compile time configuration

[

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

](RS_BRF_01136)

5.1.1.2.5 [SRS_CryptoStack_00094] The configuration files of the crypto stack modules shall be readable for human beings

[

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

](RS_BRF_01456)

5.1.1.3 Initialization

None

5.1.1.4 Normal Operation

5.1.1.4.1 [SRS_CryptoStack_00009] The Crypto Stack shall support reentrancy for all crypto services

[

Type:	Valid
Description:	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.
Rationale:	Crypto jobs shall be processable simultaneously

Use Case:	Different applications may use cryptographic services in parallel. Handling of different tasks at the same time is necessary.
Dependencies:	--
Supporting Material:	--

](RS_BRF_02033)

5.1.1.4.2 [SRS_CryptoStack_00010] The Crypto Stack shall conceal symmetric keys from the users of crypto services

[

Type:	Valid
Description:	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.
Rationale:	If keys are stored in the application, this increases the chances of invalidation of keys or keys being compromised.
Use Case:	Keys residing in the HSM
Dependencies:	--
Supporting Material:	--

](RS_BRF_02031, RS_BRF_01946)

5.1.1.4.3 [SRS_CryptoStack_00011] The Crypto Stack shall conceal asymmetric private keys from the users of Crypto services

[

Type:	Valid
Description:	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.
Rationale:	If keys are stored in the application, this increases the chances of invalidation of keys or keys being compromised.
Use Case:	Keys residing in the HSM
Dependencies:	--
Supporting Material:	--

](RS_BRF_02031, RS_BRF_01946)

5.1.1.4.4 [SRS_CryptoStack_00019] The Crypto Stack shall identify random number generation as a cryptographic primitive which can be requested to a driver

[

Type:	Valid

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

](RS_BRF_02031)

5.1.1.4.5 [SRS_CryptoStack_00020] The Crypto Stack shall identify symmetric encryption/decryption as a cryptographic primitive which can be requested to a driver

[

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

](RS_BRF_02031)

5.1.1.4.6 [SRS_CryptoStack_00021] The Crypto Stack shall identify asymmetric encryption/decryption as a cryptographic primitive which can be requested to a driver

[

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

](RS_BRF_02031)

5.1.1.4.7 [SRS_CryptoStack_00022] The Crypto Stack shall identify MAC generation/verification as a cryptographic primitive which can be requested to a driver

[

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

](RS_BRF_02031)

5.1.1.4.8 [SRS_CryptoStack_00023] The Crypto Stack shall identify asymmetric signature generation/verification as a cryptographic primitive which can be requested to a driver

[

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

](RS_BRF_02031)

5.1.1.4.9 [SRS_CryptoStack_00024] The Crypto Stack shall identify hash calculation as a cryptographic primitive which can be requested to a driver

[

Type:	Valid
Description:	The Crypto Stack shall identify hash calculation as a cryptographic primitive which can be requested to a driver.
Rationale:	Hash algorithms residing on different crypto drivers should be accessed using a homogenous interface.

Use Case:	Signature verification
Dependencies:	--
Supporting Material:	--

](RS_BRF_02031)

5.1.1.4.10 [SRS_CryptoStack_00026] The Crypto Stack shall provide an interface for the generation of asymmetric keys

[

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

](RS_BRF_02031)

5.1.1.4.11 [SRS_CryptoStack_00027] The Crypto Stack shall provide an interface for the generation of symmetric keys

[

Type:	Valid
Description:	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.
Rationale:	Key generation services residing on different Crypto drivers should be accessed using a homogenous interface.
Use Case:	Password-based key input
Dependencies:	--
Supporting Material:	--

](RS_BRF_02031)

5.1.1.4.12 [SRS_CryptoStack_00103] The Crypto Stack shall provide an interface for the derivation of symmetric keys

[

Type:	Valid
Description:	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.

Rationale:	Key derivation services residing on different Crypto drivers should be accessed using a homogenous interface.
Use Case:	Password-based key input
Dependencies:	--
Supporting Material:	--

](RS_BRF_02031)

5.1.1.4.13 [SRS_CryptoStack_00028] The Crypto Stack shall provide an interface for key exchange mechanisms

[

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

](RS_BRF_02031)

5.1.1.4.14 [SRS_CryptoStack_00029] The Crypto Stack shall provide an interface for key wrapping/extraction mechanisms

[

Type:	Valid
Description:	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.
Rationale:	Key wrapping and encapsulation algorithms implemented in the driver shall not be accessed directly by the users and need to be abstracted.
Use Case:	Session handling
Dependencies:	--
Supporting Material:	--

](RS_BRF_02031)

5.1.1.4.15 [SRS_CryptoStack_00030] The Crypto Stack shall identify a secure counter as cryptographic primitive with various operations

[

Type:	Valid
Description:	The Crypto Stack shall provide interface to the Crypto Driver for secure counter. The following operations shall be possible on a counter: 1. read counter 2. increment counter Note: “Secure” in the context of Secure Counter means, that the counter shall be secured against direct access of the user. Thus, for a meaningful implementation, the Secure Counter should be resident in a special secured nonvolatile ROM (e.g. in an HSM)
Rationale:	Counters maintained by the driver shall not be accessed directly by the users and need to be abstracted.
Use Case:	Keep a counter in secured areas of an HSM
Dependencies:	--
Supporting Material:	--

](RS_BRF_01946)

5.1.1.4.16 [SRS_CryptoStack_00031] The Crypto Stack shall provide an interface for parsing certificates

[

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

](RS_BRF_01946)

5.1.1.4.17 [SRS_CryptoStack_00061] The Crypto Stack shall support detection of invalid keys

[

Type:	Valid
Description:	The implementation of a cryptographic primitive shall detect and reject invalid keys.
Rationale:	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.
Use Case:	RSA and several Elliptic Curve Cryptosystems
Dependencies:	--

Supporting Material:	--

](RS_BRF_02031, RS_BRF_01946)

5.1.1.5 Shutdown Operation

None

5.1.1.6 Fault Operation

None

5.1.2 Crypto Service Manager

5.1.2.1 General

5.1.2.1.1 [SRS_CryptoStack_00006] Each primitive of the CRYIF shall belong to exactly one service of the CSM

[

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

](RS_BRF_02032)

5.1.2.2 Configuration

5.1.2.2.1 [SRS_CryptoStack_00079] The job processing mode (synchronous or asynchronous) of a CSM service shall be defined by static configuration

[

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

](RS_BRF_01456, RS_BRF_01136)

5.1.2.2.2 [SRS_CryptoStack_00102] The priority of a user and its crypto jobs shall be defined by static configuration

[

Type:	Valid
Description:	The user's priority shall be defined by static configuration. All jobs of that user inherit that priority.

Rationale:	There are crypto jobs which have to 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.
Use Case:	Prioritized job processing
Dependencies:	--
Supporting Material:	--

]()

5.1.2.2.3 [SRS_CryptoStack_00080] The set of cryptographic services provided by the CSM shall be defined by static configuration

[

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

](RS_BRF_01456, RS_BRF_01136)

5.1.2.2.4 [SRS_CryptoStack_00081] The CSM module specification shall specify which other modules are required

[

Type:	Valid
Description:	The CSM module specification shall specify which other modules are required.
Rationale:	Basic functionality
Use Case:	--
Dependencies:	--
Supporting Material:	--

](RS_BRF_01456, RS_BRF_01064)

5.1.2.2.5 [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

[

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

](RS_BRF_01456, RS_BRF_01064)

5.1.2.3 Initialization

5.1.2.4 Normal Operation

5.1.2.4.1 [SRS_CryptoStack_00084] The CSM module shall use the streaming approach for some selected services

[

Type:	Valid
Description:	The CSM module shall use the streaming approach for some provided services (see Software Specification of CSM).
Rationale:	Basic functionality
Use Case:	It shall be possible to hand over the input data in small chunks to the service.
Dependencies:	--
Supporting Material:	

](RS_BRF_01456)

5.1.2.5 Shutdown Operation

<what to do when the module is shut down>

5.1.2.6 Fault Operation

5.1.2.6.1 [SRS_CryptoStack_00086] The CSM module shall distinguish between error types

[

Type:	Valid
Description:	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
Rationale:	Basic functionality
Use Case:	--
Dependencies:	--
Supporting Material:	--

](RS_BRF_02168, RS_BRF_02272)

5.1.2.6.2 [SRS_CryptoStack_00087] The CSM module shall report detected development errors to the Development Error Tracer

[

Type:	Valid
Description:	The CSM module shall report detected development errors to the Development Error Tracer
Rationale:	Basic functionality
Use Case:	--
Dependencies:	--
Supporting Material:	--

](RS_BRF_02168, RS_BRF_02272)

5.1.2.6.3 [SRS_CryptoStack_00087] The CSM module shall not return specific development error codes via the API

[

Type:	Valid

Description:	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.
Rationale:	Basic functionality
Use Case:	--
Dependencies:	--
Supporting Material:	--

](RS_BRF_00129, RS_BRF_02168)

5.1.2.6.4 [SRS_CryptoStack_00088] The CSM shall check passed API parameters for validity

[

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

](RS_BRF_00129, RS_BRF_02168, RS_BRF_02232)

5.1.3 Crypto Interface

5.1.3.1 Configuration

5.1.3.1.1 [SRS_CryptoStack_00014] The Crypto Interface shall have an interface to the static configuration information of the Crypto Driver

[

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

](RS_BRF_01008)

5.1.3.1.2 [SRS_CryptoStack_00015] Channels mapped to different Crypto Driver Objects shall be uniquely configurable in Crypto Interface

[

Type:	Valid

Description:	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.
Rationale:	Each crypto driver object in the driver can be abstracted from and utilized by CSM using virtual channels.
Use Case:	Two hardware resources: one for symmetric cryptography, one for asymmetric cryptography
Dependencies:	--
Supporting Material:	--

](RS_BRF_02032, RS_BRF_01456, RS_BRF_01136)

5.1.3.2 Initialization

5.1.3.3 Normal Operation

5.1.3.4 Shutdown Operation

None

5.1.3.5 Fault Operation

5.1.3.5.1 [SRS_CryptoStack_00034] The Crypto Interface shall report detected development errors to the Default Error Tracer

[

Type:	Valid
Description:	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.
Rationale:	Debugging Support
Use Case:	All the input parameters and internal states have to be validated before processing.
Dependencies:	--
Supporting Material:	--

](RS_BRF_02232)

5.1.4 Crypto Driver

5.1.4.1 Configuration

5.1.4.1.1 [SRS_CryptoStack_00036] The Crypto Driver shall allow static configuration of Crypto Driver Objects

[

Type:	Valid
Description:	The Crypto Driver shall allow defining of different Crypto Driver Objects.
Rationale:	Abstraction of different hardware units
Use Case:	Parallel processing of symmetric operations and asymmetric operations
Dependencies:	--
Supporting Material:	--

](RS_BRF_01456, RS_BRF_01136)

5.1.4.1.2 [SRS_CryptoStack_00038] The Crypto Driver shall allow configuration of parameters related to secure counter

[

Type:	Valid
Description:	The Crypto Driver shall provide configuration parameters with respect to Secure Counter which allow to decide globally the maximum number of Secure Counters supported.
Rat/ionale:	Making such parameters configurable defines a limit with respect to the Hardware resources available so the User cannot avail more resources than configured.
Use Case:	Usage of different counters for different purposes.
Dependencies:	--
Supporting Material:	--

](RS_BRF_01946)

5.1.4.1.3 [SRS_CryptoStack_00104] Crypto Interface keys mapped to different Crypto Driver Keys shall be uniquely configurable in the Crypto Interface

[

Type:	Valid
Description:	The Crypto Interface shall support a configuration model where all CRYIF keys shall be statically mapped to keys in the crypto driver.
Rationale:	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.
Use Case:	Multiple Crypto Driver modules where each module has its own key identifiers

Dependencies:	--
Supporting Material:	--

](RS_BRF_02032, RS_BRF_01456, RS_BRF_01136)

5.1.4.2 Initialization

5.1.4.3 Normal Operation

5.1.4.3.1 [SRS_CryptoStack_00098] The Crypto Driver shall provide access to all cryptographic algorithms supported by the hardware

[

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

]()

5.1.4.4 Shutdown Operation

None

5.1.4.5 Fault Operation

5.2 Non-Functional Requirements (Qualities)

5.2.1 General

5.2.2 Crypto Service Manager

5.2.2.1 [SRS_CryptoStack_00088] The CSM module shall provide an abstraction layer which offers a standardized interface to higher software layers to access cryptographic algorithms

[

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

](RS_BRF_01456, RS_BRF_01016, RS_BRF_01056)

5.2.2.2 [SRS_CryptoStack_00089] The CSM module shall be located in the AUTOSAR service layer

[

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

](RS_BRF_01016, RS_BRF_01408)

5.2.2.3 [SRS_CryptoStack_00090] The CSM shall provide an interface to be accessible via the RTE

[

Type:	Valid
Description:	The CSM shall provide an interface to be accessible via the RTE.
Rationale:	The CSM module shall be accessible from applications above the RTE.
Use Case:	Applications which require crypto services.
Dependencies:	--
Supporting Material:	--

](RS_BRF_01408, RS_BRF_01280)

5.2.2.4 [SRS_CryptoStack_00091] The CSM shall provide one Provide--Port for each configuration

[

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

](RS_BRF_01056, RS_BRF_01280, RS_BRF_01408, RS_BRF_01456)

5.2.2.5 [SRS_CryptoStack_00092] The CSM shall provide one Require-Port for each configuration

[

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

](RS_BRF_01056, RS_BRF_01280, RS_BRF_01408, RS_BRF_01456)

5.2.3 Crypto Interface

5.2.3.1 [SRS_CryptoStack_00075] The Crypto Interface shall be the interface layer between the underlying crypto driver(s) and upper layers

[

Type:	Valid
Description:	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
Rationale:	Interfaces and interaction
Use Case:	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.
Dependencies:	--
Supporting Material:	AUTOSAR_WP Architecture_SoftwareArchitecture

](RS_BRF_01000, RS_BRF_01008, RS_BRF_01016)

5.2.3.2 [SRS_CryptoStack_00076] The Crypto Interface implementation and interface shall be independent from underlying Crypto Hardware or Software

[

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

](RS_BRF_01000, RS_BRF_02031)

5.2.4 Crypto Driver

5.2.4.1 [SRS_CryptoStack_00095] The Crypto Driver module shall strictly separate error and status information

[

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

](RS_BRF_00129, RS_BRF_02168, RS_BRF_02232, RS_BRF_02272)

6 Requirements Tracing

Requirement	Description	Satisfied by
RS_BRF_00129	AUTOSAR shall support data corruption detection and protection	SRS_CryptoStack_00087, SRS_CryptoStack_00088, SRS_CryptoStack_00095
RS_BRF_01000	AUTOSAR architecture shall organize the BSW in a hardware independent and a hardware dependent layer	SRS_CryptoStack_00075, SRS_CryptoStack_00076
RS_BRF_01008	AUTOSAR shall organize the hardware dependent layer in a microcontroller independent and a microcontroller dependent layer	SRS_CryptoStack_00014, SRS_CryptoStack_00075
RS_BRF_01016	AUTOSAR shall provide a modular design inside software layers	SRS_CryptoStack_00075, SRS_CryptoStack_00088, SRS_CryptoStack_00089
RS_BRF_01056	AUTOSAR BSW modules shall provide standardized interfaces	SRS_CryptoStack_00088, SRS_CryptoStack_00091, SRS_CryptoStack_00092
RS_BRF_01064	AUTOSAR BSW shall provide callback functions in order to access upper layer modules	SRS_CryptoStack_00081, SRS_CryptoStack_00082
RS_BRF_01136	AUTOSAR shall support variants of configured BSW data resolved after system start-up	SRS_CryptoStack_00013, SRS_CryptoStack_00015, SRS_CryptoStack_00036, SRS_CryptoStack_00079, SRS_CryptoStack_00080, SRS_CryptoStack_00104
RS_BRF_01280	AUTOSAR RTE shall offer the external interfaces between Software Components and between Software Components and BSW	SRS_CryptoStack_00090, SRS_CryptoStack_00091, SRS_CryptoStack_00092
RS_BRF_01408	AUTOSAR shall provide a service layer that is accessible from each basic software layer	SRS_CryptoStack_00089, SRS_CryptoStack_00090, SRS_CryptoStack_00091, SRS_CryptoStack_00092
RS_BRF_01456	AUTOSAR services shall provide system wide cryptographic functionality	SRS_CryptoStack_00007, SRS_CryptoStack_00015, SRS_CryptoStack_00036, SRS_CryptoStack_00079, SRS_CryptoStack_00080, SRS_CryptoStack_00081, SRS_CryptoStack_00082, SRS_CryptoStack_00084, SRS_CryptoStack_00088, SRS_CryptoStack_00091, SRS_CryptoStack_00092, SRS_CryptoStack_00094, SRS_CryptoStack_00100, SRS_CryptoStack_00101, SRS_CryptoStack_00104
RS_BRF_01946	AUTOSAR microcontroller abstraction shall provide	SRS_CryptoStack_00008, SRS_CryptoStack_00010, SRS_CryptoStack_00011, SRS_CryptoStack_00030, SRS_CryptoStack_00031, SRS_CryptoStack_00038,

	access to cryptographic hardware	SRS_CryptoStack_00061, SRS_CryptoStack_00105
RS_BRF_02031	AUTOSAR shall provide uniform access to cryptographic solutions implemented either by software or hardware	SRS_CryptoStack_00007, SRS_CryptoStack_00008, SRS_CryptoStack_00010, SRS_CryptoStack_00011, SRS_CryptoStack_00019, SRS_CryptoStack_00020, SRS_CryptoStack_00021, SRS_CryptoStack_00022, SRS_CryptoStack_00023, SRS_CryptoStack_00024, SRS_CryptoStack_00026, SRS_CryptoStack_00027, SRS_CryptoStack_00028, SRS_CryptoStack_00029, SRS_CryptoStack_00061, SRS_CryptoStack_00076, SRS_CryptoStack_00103, SRS_CryptoStack_00105
RS_BRF_02032	AUTOSAR security shall allow integration of cryptographic primitives into the cryptographic service manager	SRS_CryptoStack_00003, SRS_CryptoStack_00006, SRS_CryptoStack_00015, SRS_CryptoStack_00104
RS_BRF_02033	AUTOSAR shall provide concurrent access to cryptographic services	SRS_CryptoStack_00009
RS_BRF_02168	AUTOSAR diagnostics shall provide a central classification and handling of abnormal operative conditions	SRS_CryptoStack_00086, SRS_CryptoStack_00087, SRS_CryptoStack_00088, SRS_CryptoStack_00095
RS_BRF_02232	AUTOSAR shall support development with run-time assertion checks	SRS_CryptoStack_00034, SRS_CryptoStack_00088, SRS_CryptoStack_00095
RS_BRF_02272	AUTOSAR shall offer tracing of application software behavior	SRS_CryptoStack_00086, SRS_CryptoStack_00087, SRS_CryptoStack_00095

7 References

7.1 Deliverables of AUTOSAR

[1] General Requirements on Basic Software Modules
AUTOSAR_SRS_BSWGeneral.pdf

[2] Layered Software Architecture
AUTOSAR_EXP_LayeredSoftwareArchitecture.pdf

[3] Software Standardization Template
AUTOSAR_TPS_StandardizationTemplate.pdf

[4] Specification of System Template
AUTOSAR_TPS_SystemTemplate.pdf

[5] Requirements on AUTOSAR Features
AUTOSAR_RS_Features

7.2 Related standards and norms