

MICROSAR CRYPTO

Technical Reference

CRYPTO CRYWRAPPER

Version 2.1.0

Authors	Tobias Finke
Status	Released

Document Information

History

Author	Date	Version	Remarks
Tobias Finke	2016-12-21	1.00.00	Initial creation
Tobias Finke	2017-10-06	2.01.00	Release of component

Reference Documents

No.	Source	Title	Version
[1]	AUTOSAR	AUTOSAR_CryptoDriver.pdf	4.3.0
[2]	AUTOSAR	AUTOSAR_SWS_DET.pdf	4.3.0
[3]	AUTOSAR	AUTOSAR_TR_BSWModuleList.pdf	4.3.0
[4]	HIS	2009-04-01 SHE Functional Specification v1.1 (rev439)	1.1

Contents

1	Component History	6
2	Introduction.....	7
2.1	Architecture Overview	7
3	Functional Description	9
3.1	Features	9
3.1.1	Deviations	9
3.1.2	Additions/ Extensions.....	9
3.1.3	Limitations.....	10
3.1.3.1	Cryptographic Algorithms and Modes	10
3.1.3.2	Certificate Handling.....	10
3.1.3.3	Key Generation.....	10
3.1.3.4	AEAD Det Checks	10
3.1.3.5	Asynchronous CRY.....	10
3.2	Initialization	10
3.3	Main Functions	10
3.4	SHE Key Update Protocol.....	10
3.4.1	Preconditions	10
3.5	Error Handling.....	11
3.5.1	Development Error Reporting.....	11
3.6	Key Management.....	12
3.6.1	Custom Key Elements.....	12
3.6.2	Import Keys.....	13
3.6.3	Export Keys	14
3.7	Algorithm Parameter Overview	14
3.8	Crypto Driver Configuration.....	15
3.8.1	General Configuration.....	15
3.8.2	Configuration of Keys.....	15
3.8.2.1	Key Elements.....	15
3.8.2.2	Key Types.....	16
3.8.2.3	Keys	16
3.8.3	Wrapping of CRY	17
3.8.3.1	Configuration of the underlying CRY	17
3.8.3.2	Algorithms.....	17
3.8.3.3	Key Functions.....	19
4	Integration.....	20

4.1	Scope of Delivery	20
4.1.1	Static Files	20
4.1.2	Dynamic Files	20
4.2	Critical Sections	21
5	API Description	22
5.1	Services provided by CRYPTO	22
5.1.1	Crypto_30_CryWrapper_Init.....	22
5.1.2	Crypto_30_CryWrapper_InitMemory	22
5.1.3	Crypto_30_CryWrapper_GetVersionInfo	23
5.1.4	Crypto_30_CryWrapper_ProcessJob	23
5.1.5	Crypto_30_CryWrapper_CancelJob.....	24
5.2	Key Management Functions	25
5.2.1	Crypto_30_CryWrapper_KeyCopy	25
5.2.2	Crypto_30_CryWrapper_KeyElementCopy	25
5.2.3	Crypto_30_CryWrapper_KeyElementIdsGet	26
5.2.4	Crypto_30_CryWrapper_KeyElementSet	27
5.2.5	Crypto_30_CryWrapper_KeyValidSet	28
5.2.6	Crypto_30_CryWrapper_KeyElementGet.....	28
5.2.7	Crypto_30_CryWrapper_RandomSeed	29
5.2.8	Crypto_30_CryWrapper_KeyGenerate.....	30
5.2.9	Crypto_30_CryWrapper_KeyDerive	31
5.2.10	Crypto_30_CryWrapper_KeyExchangeCalcPubVal	31
5.2.11	Crypto_30_CryWrapper_KeyExchangeCalcSecret	32
5.2.12	Crypto_30_CryWrapper_CertificateParse	33
5.2.13	Crypto_30_CryWrapper_CertificateVerify.....	34
5.3	Services used by CRYPTO	34
6	Configuration.....	36
6.1	Configuration Variants.....	36
7	Glossary and Abbreviations	37
7.1	Glossary	37
7.2	Abbreviations	37
8	Contact.....	38

Illustrations

Figure 2-1	AUTOSAR 4.2 Architecture Overview	7
Figure 2-2	Interfaces to adjacent modules of the CRYPTO	8
Figure 3-1	Example SymKeyExtract Configuration for Vector CRY	18
Figure 3-2	Example SymKeyExtract configuration for third party CRY	19

Tables

Table 1-1	Component history.....	6
Table 3-1	Supported AUTOSAR standard conform features	9
Table 3-2	Not supported AUTOSAR standard conform features	9
Table 3-3	Features provided beyond the AUTOSAR standard	9
Table 3-4	Service IDs	11
Table 3-5	Errors reported to DET	12
Table 3-6	Supported data format for key import.....	13
Table 3-7	Supported data format for key export.....	14
Table 3-8	Overview of the required algorithm parameter	15
Table 3-9	Description of preconfigured key elements	16
Table 3-10	Description of preconfigured key types	16
Table 3-11	Description of pre-configured keys.....	17
Table 4-1	Static files	20
Table 4-2	Generated files	20
Table 5-1	Crypto_30_CryWrapper_Init	22
Table 5-2	Crypto_30_CryWrapper_InitMemory.....	23
Table 5-3	Crypto_30_CryWrapper_GetVersionInfo.....	23
Table 5-4	Crypto_30_CryWrapper_ProcessJob.....	24
Table 5-5	Crypto_30_CryWrapper_CancelJob	25
Table 5-6	Crypto_30_CryWrapper_KeyCopy.....	25
Table 5-7	Crypto_30_CryWrapper_KeyElementCopy.....	26
Table 5-8	Crypto_30_CryWrapper_KeyElementIdsGet.....	27
Table 5-9	Crypto_30_CryWrapper_KeyElementSet.....	28
Table 5-10	Crypto_30_CryWrapper_KeyValidSet	28
Table 5-11	Crypto_30_CryWrapper_KeyElementGet	29
Table 5-12	Crypto_30_CryWrapper_RandomSeed	30
Table 5-13	Crypto_30_CryWrapper_KeyGenerate	30
Table 5-14	Crypto_30_CryWrapper_KeyDerive.....	31
Table 5-15	Crypto_30_CryWrapper_KeyExchangeCalcPubVal	32
Table 5-16	Crypto_30_CryWrapper_KeyExchangeCalcSecret.....	33
Table 5-17	Crypto_30_CryWrapper_CertificateParse.....	34
Table 5-18	Crypto_30_CryWrapper_CertificateVerify	34
Table 5-19	Services used by the CRYPTO	34
Table 7-1	Glossary	37
Table 7-2	Abbreviations.....	37

1 Component History

The component history gives an overview over the important milestones that are supported in the different versions of the component.

Component Version	New Features
1.00.00	Initial beta release
2.00.00	Redesign of component
2.01.00	Safe Bsw Release of Component Added several configurations options to support Cry implementations of different vendors

Table 1-1 Component history

2 Introduction

This document describes the functionality, API and configuration of the AUTOSAR BSW module CRYPTO as specified in [1].

Supported AUTOSAR Release*:	4.3	
Supported Configuration Variants:	pre-compile	
Vendor ID:	CRYPTO_VENDOR_ID	30 decimal (= Vector-Informatik, according to HIS)
Module ID:	CRYPTO_MODULE_ID	114 decimal (according to ref. [3])

* For the detailed functional specification please also refer to the corresponding AUTOSAR SWS.

The Crypto Driver (CRYPTO) is called by the Crypto Interface (CRYIF) and performs the specific cryptographic functionality. The CRYPTO specification [1] offers a superset of algorithms which can be extended by 'custom algorithms'. This software-based Crypto Driver offers a subset of algorithms and features which is described in 3.1.

2.1 Architecture Overview

The following figure shows where the CRYPTO is located in the AUTOSAR architecture.

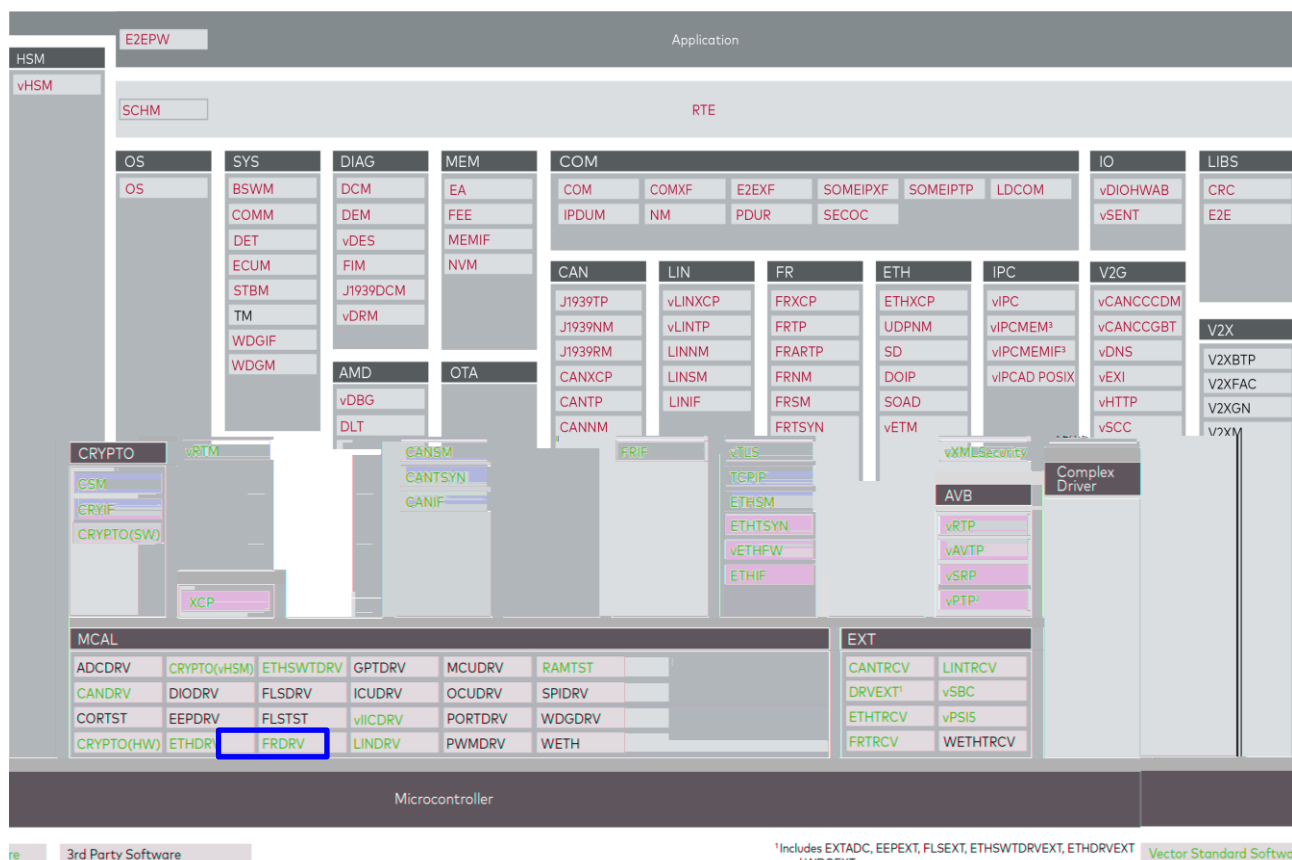


Figure 2-1 AUTOSAR 4.2 Architecture Overview

The next figure shows the interfaces to adjacent modules of the CRYPTO. These interfaces are described in chapter 5.

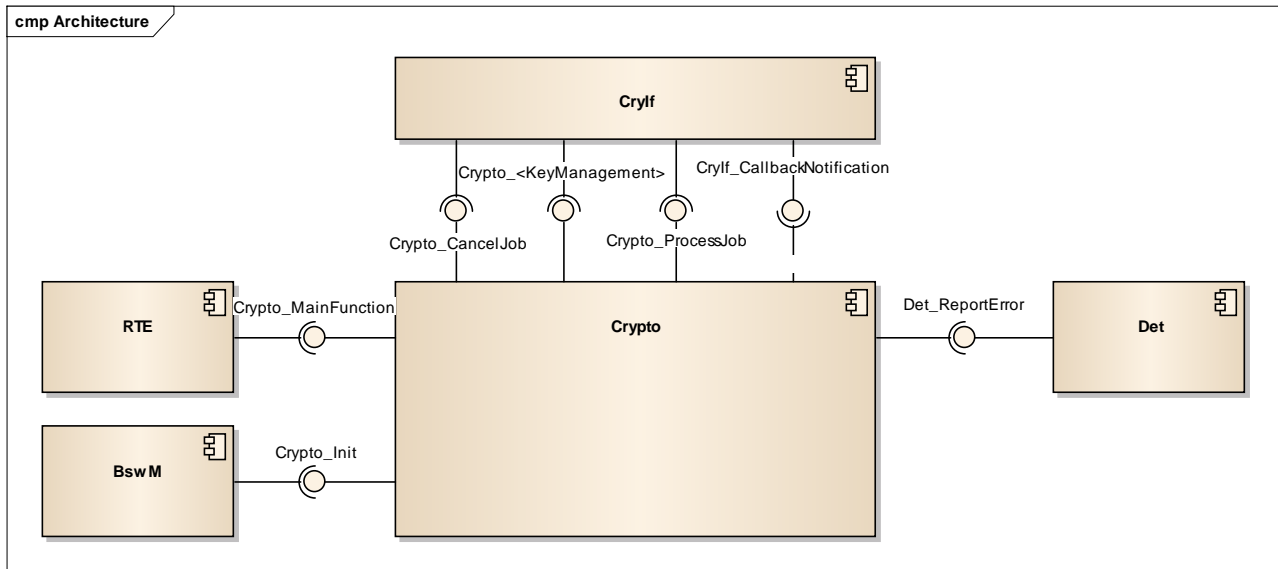


Figure 2-2 Interfaces to adjacent modules of the CRYPTO

3 Functional Description

3.1 Features

The features listed in the following tables cover the complete functionality specified for the CRYPTO.

The AUTOSAR standard functionality is specified in [1], the corresponding features are listed in the tables

- > Table 3-1 Supported AUTOSAR standard conform features
- > Table 3-2 Not supported AUTOSAR standard conform features

Vector Informatik provides further CRYPTO functionality beyond the AUTOSAR standard. The corresponding features are listed in the table

- > Table 3-3 Features provided beyond the AUTOSAR standard

The following features specified in [1] are supported:

Supported AUTOSAR Standard Conform Features
Encrypt/Decrypt: AES-128 ECB/CBC
MAC: AES CMAC
RNG: TRNG

Table 3-1 Supported AUTOSAR standard conform features

3.1.1 Deviations

The following features specified in [1] are not supported:

Not Supported AUTOSAR Standard Conform Features
Certificate handling
All algorithms not stated in Table 3-1
The Read/Write Access for Key Elements are not checked when they are not located inside the SHE.

Table 3-2 Not supported AUTOSAR standard conform features

3.1.2 Additions/ Extensions

The following features are provided beyond the AUTOSAR standard:

Features Provided Beyond The AUTOSAR Standard
Storage of Keys into the SHE with the SymKeyExtract-API provided by the underlying CRY by using the 'SHE Key Update Protocol' [4]
Export of Keys from the SHE with the SymKeyWrapSym-API provided by the underlying CRY by using the 'SHE Key Update Protocol' [4]
Forwarding and converting service requests to the ASR4.2 CRY APIs.

Table 3-3 Features provided beyond the AUTOSAR standard

3.1.3 Limitations

3.1.3.1 Cryptographic Algorithms and Modes

Only a subset of the stated algorithms and modes in the AUTOSAR SWS [1] are currently supported. The list of algorithms is described in Table 3-1.

3.1.3.2 Certificate Handling

Currently there is no implementation to parse and verify certificates. However the API is still available for compatibility reasons.

3.1.3.3 Key Generation

Currently there is no implementation to generate keys. However the API is still available for compatibility reasons.

3.1.3.4 AEAD Det Checks

The Det checks for AEAD differ from the table in [1]. There is a different handling required for the AEAD implementation. The parameter check is described in Table 3-8.

3.1.3.5 Asynchronous CRY

Handling of a CRY which is configured asynchronous is not supported.

3.2 Initialization

Before any other functionality of the CRYPTO module can be called the initialization function `Crypto_30_CryWrapper_Init()` has to be called by the BSWM.

For manual null initialization of RAM variables the CRYPTO offers the function `Crypto_30_CryWrapper_InitMemory()` which can be called before the `Crypto_30_CryWrapper_Init()`.

3.3 Main Functions

The CRYPTO module implementation provides one main function. When the usage of asynchronous job processing is enabled, this main function has to be called cyclically on task level. The main function is responsible to start processing of the job.

3.4 SHE Key Update Protocol

The MICROSAR CRYPTO has the ability to update a key in the SHE hardware by using the SHE Key Update Protocol [4].

3.4.1 Preconditions

When a key shall be updated using the SHE Key Update Protocol the

- > Write Access has to be Encrypted
- > Key material must be a 64 byte long concatenated M1|M2|M3 message

- > If M4 and M5 are needed the “proof” key element must be available and has to be of size 48 bytes

3.5 Error Handling

3.5.1 Development Error Reporting

By default, development errors are reported to the DET using the service `Det_ReportError()` as specified in [2], if development error reporting is enabled (i.e. pre-compile parameter `CRYPTO_DEV_ERROR_REPORT==STD_ON`).

If another module is used for development error reporting, the function prototype for reporting the error can be configured by the integrator, but must have the same signature as the service `Det_ReportError()`.

The reported CRYPTO ID is 114.

The reported service IDs identify the services which are described in 5.1. The following table presents the service IDs and the related services:

Service ID	Service
0x00	Crypto_30_CryWrapper_Init()
0x01	Crypto_30_CryWrapper_GetVersionInfo()
0x03	Crypto_30_CryWrapper_ProcessJob()
0x0E	Crypto_30_CryWrapper_CancelJob()
0x04	Crypto_30_CryWrapper_KeyElementSet()
0x05	Crypto_30_CryWrapper_KeyValidSet()
0x06	Crypto_30_CryWrapper_KeyElementGet()
0x0F	Crypto_30_CryWrapper_KeyElementCopy()
0x10	Crypto_30_CryWrapper_KeyCopy()
0x11	Crypto_30_CryWrapper_KeyElementIdsGet()
0x0D	Crypto_30_CryWrapper_RandomSeed()
0x07	Crypto_30_CryWrapper_KeyGenerate()
0x08	Crypto_30_CryWrapper_KeyDerive()
0x09	Crypto_30_CryWrapper_KeyExchangeCalcPubVal()
0x0A	Crypto_30_CryWrapper_KeyExchangeCalcSecret()
0x0B	Crypto_30_CryWrapper_CertificateParse()
0x12	Crypto_30_CryWrapper_CertificateVerify()
0x0C	Crypto_30_CryWrapper_MainFunction()

Table 3-4 Service IDs

The errors reported to DET are described in the following table:

Error Code	Description
0x00	CRYPTO_E_UNINIT
0x01	CRYPTO_E_INIT_FAILED

Error Code	Description
0x02	CRYPTO_E_PARAM_POINTER
0x04	CRYPTO_E_PARAM_HANDLE
0x05	CRYPTO_E_PARAM_VALUE

Table 3-5 Errors reported to DET

3.6 Key Management

This hardware driver has some special features regarding the key management functionalities. These features are described in the following.

3.6.1 Custom Key Elements

The crypto driver does not use any custom key elements.

3.6.2 Import Keys

For storing keys inside the SHE, the API `Crypto_30_CryWrapper_KeyElementSet()` is used. Table 3-6 shows which input length and key group mapping is necessary to import a specific type of key.

Input	Length	Key Ids
-------	--------	---------

3.6.3 Export Keys

For exporting keys from the HSM, the API `Crypto_30_CryWrapper_KeyElementGet()` is used. Table 3-7 shows which length and key group mapping is necessary to export a specific type of key.

Output	Length	Key Id
SHE RAM Key as M1-M5 The key data consists of the concatenation of M1 M2 M3 M4 M5.	112	Key_Ram (0x0E)

Table 3-7 Supported data format for key export

3.7 Algorithm Parameter Overview

The required algorithm parameters are described in the following table. The required parameter differs from the table in [1].

Member													
Service	inputPtr	inputLength	secondaryInputPtr	secondaryInputLength	tertiaryInputPtr	tertiaryInputLength	outputPtr	outputLengthPtr	secondaryOutputPtr	secondaryOutputLengthPtr	verifyPtr	output64Ptr	mode
HASH	U	U					F	F					SUF
MACGENERATE	U	U					F	F					SUF
MACVERIFY	U	U	F	F							F		SUF
ENCRYPT	U	U					UF	UF					SUF
DECRYPT	U	U					UF	UF					SUF
AEADENCRYPT	U	U	V	U~			UF	UF	F	F			SUF
AEADDECRYPT	U	U	V	U~	F	F	UF	UF			F		SUF
SIGNATUREGENERATE	UF	U F~					F	F					SUF
SIGNATUREVERIFY	UF	U F~	F	F							F		SUF
SECCOUNTERINCREMENT													
SECCOUNTERREAD												F	
RANDOMGENERATE							F	F					

S: member required in Start mode.
 U: member required in Update mode.
 V: member optional in Update mode.
 F: member required in Finish mode.
 ~: no Det check required / 0 is a valid value.

Table 3-8 Overview of the required algorithm parameter

3.8 Crypto Driver Configuration

The Crypto Driver comes with pre-configured and recommended settings which define the supported algorithm primitives and its key with the associated key elements.

3.8.1 General Configuration

In the CryptoWrapperGeneral configuration section, the specifics of the underlying CRY can be defined. Refer to the description in the online help of the DaVinci CFG5 for the parameter descriptions and their values for different driver vendors.

3.8.2 Configuration of Keys

The main configuration part for the keys is provided by the pre-configuration and is not editable by the user.

3.8.2.1 Key Elements

There five basic key elements which are needed regarding the HW CRYs.

Key Element	Description
Crypto_30_CryWrapper_AesIV	This Key element is used to store the init vector of an AES operation. The IV is only needed if the block mode of AES is CBC. It is 16 Bytes long and can be modified with the Csm_KeyElementSet() and Csm_KeyElementGet().
Crypto_30_CryWrapper_SheNvmKey	This key element is used as placeholder for a nonvolatile key slot in the SHE. It has a size of 64 Bytes which will be used for the concatenation of M1, M2 and M3. It can only be written with the function Csm_KeyElementSet().
Crypto_30_CryWrapper_SheProof	This key element is used to store the Proof (M4 M5) after a successful Key Update protocol of the SHE. It has a size of 48 Bytes which will be used for the concatenation of M4 and M5. It can be read with Csm_KeyElementGet().

Crypto_30_CryWrapper_SheRamKey	This key element is used as placeholder for the RAM key slot of the SHE. It can be set with Csm_KeyElementSet().
Crypto_30_CryWrapper_CryptoKey	This key element is used to store a key in the key storage of the crypto driver. When an algorithm uses this key element, the key material is automatically stored to the RAM key slot of the SHE. It is 16 Bytes long and can be modified (even partially) with Csm_KeyElementSet() and Csm_KeyElementGet().

Table 3-9 Description of preconfigured key elements

3.8.2.2 Key Types

Key Type	Description
SheNvmKey	A key of this type represents a nonvolatile SHE keyslot (e.g. KEY_1) without the ability to compute AES CBC or store the result of the key update protocol.
SheRamKey	A key of this type represents a SHE RAM keyslot (KEY_RAM) without the ability to compute AES CBC or store the result of the key update protocol.
<...>IV	The key of this type has the ability to store the IV which is needed for AES in CBC mode.
<...>Proof<...>	The key of this type has the ability to store the M4 and M5 which is generated when a key is successfully updated. M4 and M5 can be used to verify the key update process.
CryptoKey	A key of this key type is stored in the key storage of the CRYPTO and loaded into the SHE RAM-Keyslot when it is needed.

Table 3-10 Description of preconfigured key types

3.8.2.3 Keys

The following keys represent the keys which are available in a SHE.

They only have recommended types. This way, the user can define if for example the key has to hold an IV. In this case, reference a key type which includes the IV key element.

The Ids of the keys match the keyId of the SHE specification.

Key	Description
SecretKey	NVM key which is stored inside the SHE hardware. It is only used when exporting the Key_Ram as AuthId.
MasterEcuKey	NVM key which is stored inside the SHE hardware. It can be set by using the SHE key update protocol.
BootMac	NVM key which is stored inside the SHE hardware. It can be set by using the SHE key update protocol.
BootMacKey	NVM key which is stored inside the SHE hardware. It can be set by using the SHE key update protocol.
Key_1	NVM key which is stored inside the SHE hardware. It can be set by using the SHE key update protocol.
Key_<...>	NVM key which is stored inside the SHE hardware. It can be set by using the SHE key update protocol.
Key_20	NVM key which is stored inside the SHE hardware. It can be set by using the SHE key update protocol.
Key_Ram	RAM key which is stored inside the SHE hardware. It can be set as plaintext key and can be exported as M1M2M3M4M5
CryptoKey	Key located in the unsecure key storage on the application core (RAM). When the key is used by a job, it will be loaded automatically into Key_Ram.

Table 3-11 Description of pre-configured keys

3.8.3 Wrapping of CRY

3.8.3.1 Configuration of the underlying CRY

- All services of the underlying CRY have to match the mapping which is configured in the general configuration of the CRYPTO.
- The underlying CRY has to be configured synchronous

3.8.3.2 Algorithms

CRY services which are seen as algorithms are:

- MacGenerate
- MacVerify
- SymEncrypt
- SymDecrypt

- RandomGenerate

To add a CRY service to the wrapper there are two possibilities. Either by manually adding the include file, primitive name and init configuration structure, or by referencing a CRY service which is present in the current configuration. The referencing functionality is only available for CRYs which can be configured in CFG5.

In Figure 3-1 it can be seen, how the necessary options are automatically filled out, when a CRY service is referenced.

Additionally, the CSM Jobs have to be referenced, which shall be used in conjunction with this CRY service.



Caution

Pay special attention to the fact, that the job primitive configured in the CSM has to match the connected CRY service. There is currently no validation available, if the settings are matching together (e.g. if AES mode is set to CBC in both the CSM-Job and the CRY service)

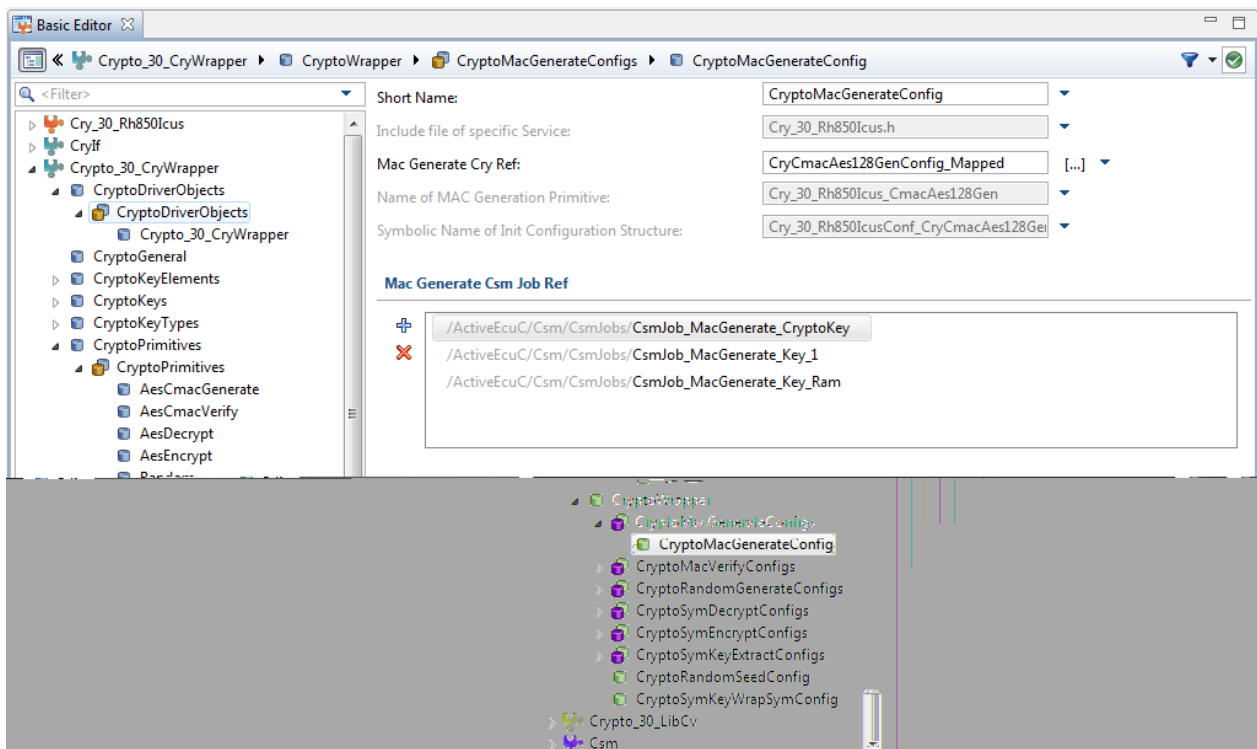


Figure 3-1 Example SymKeyExtract Configuration for Vector CRY

3.8.3.3 Key Functions

To store and export keys and feed the random number generator with a seed, different Cry interfaces are used.

For the RAM key export and the random seed functionalities, the depending CRY services have to be selected.

Only the interface for storing keys into the SHE differs slightly between vendors.

In case of a Vector CRY, just reference a CRY SymKeyExtract Service and select CRYPTO_ALL_KEYS.

If you have a CRY from a third party vendor, e.g. SHE+ for AURIX, see Figure 3-2 for an example.

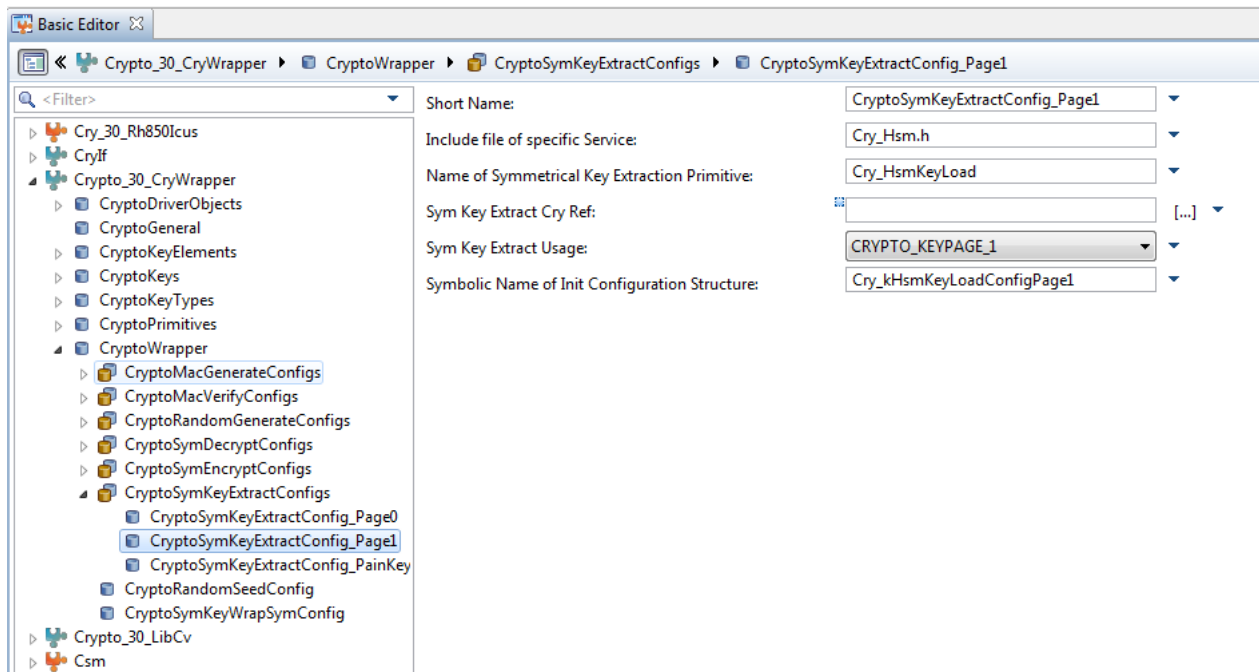


Figure 3-2 Example SymKeyExtract configuration for third party CRY

4 Integration

This chapter gives necessary information for the integration of the MICROSAR CRYPTO into an application environment of an ECU.

4.1 Scope of Delivery

The delivery of the CRYPTO contains the files which are described in the chapters 4.1.1 and 4.1.2:

4.1.1 Static Files

File Name	Description
Crypto_30_CryWrapper.c	This is the main source file of the CRYPTO
Crypto_30_CryWrapper.h	This is the source file of the CRYPTO
Crypto_30_CryWrapper_Cipher.c	This source file contains cipher algorithms
Crypto_30_CryWrapper_KeyManagement.c	This source file contains the CRYPTO's key management functions
Crypto_30_CryWrapper_KeyManagement.h	This header file contains the CRYPTO's key management functions
Crypto_30_CryWrapper_Mac.c	This source file contains MAC algorithms
Crypto_30_CryWrapper_Random.c	This source file contains PRNG algorithms
Crypto_30_CryWrapper_Services.h	This header file is used for the internal dispatcher
Crypto_30_CryWrapper_Hw.c	This source file contains the hardware dependent parts of the driver.
Crypto_30_CryWrapper_Hw.h	This header file contains the hardware dependent parts of the driver.
Crypto_30_CryWrapper_Generated_Types.h	This header file contains necessary definitions and types which must be present in the CSM. This file can be added to the CSM as an additional include.
Crypto_30_CryWrapper_Custom.h	This header file contains definitions for custom key elements. This file can be added to the CSM as an additional include.

Table 4-1 Static files

4.1.2 Dynamic Files

The dynamic files are generated by the configuration tool DaVinci Configurator 5 Pro.

File Name	Description
Crypto_30_CryWrapper_Cfg.c	This is configuration source file.
Crypto_30_CryWrapper_Cfg.h	This is configuration header file.

Table 4-2 Generated files

4.2 Critical Sections

Crypto uses the following critical sections:

> **CRYPTO_30_CRYWRAPPER_EXCLUSIVE_AREA_0**

This critical section protects workspace locking resources and so this section should never be interrupted by any other API of the Crypto module.

> **CRYPTO_30_CRYWRAPPER_EXCLUSIVE_AREA_1**

This critical section protects 32 Bit accesses and is only necessary in case of HW doesn't offer an atomic access.

> **CRYPTO_30_CRYWRAPPER_EXCLUSIVE_AREA_2**

This critical section protects the reading key access against a parallel key manipulation. The critical section should never be interrupted by any key manipulation method. So, most of the time, it is sufficient to prevent the current task to be interrupted by another task which could execute a Key Setting Method.

5 API Description

For an interfaces overview please see Figure 2-2.

5.1 Services provided by CRYPTO

5.1.1 Crypto_30_CryWrapper_Init

Prototype	
void Crypto_30_CryWrapper_Init (void)	
Parameter	
void	none
Return code	
void	none
Functional Description	
Initializes the Crypto Driver.	
Particularities and Limitations	
Specification of module initialization > Interrupts are disabled. Module is uninitialized. This function initializes the module Crypto_30_CryWrapper. It initializes all variables and sets the module state to initialized.	
Call context	
> TASK > This function is Synchronous > This function is Non-Reentrant	

Table 5-1 Crypto_30_CryWrapper_Init

5.1.2 Crypto_30_CryWrapper_InitMemory

Prototype	
void Crypto_30_CryWrapper_InitMemory (void)	
Parameter	
void	none
Return code	
void	none
Functional Description	
The function initializes variables, which cannot be initialized with the startup code.	

Particularities and Limitations
Module is uninitialized. Initialize component variables at power up.
Call context
> TASK > This function is Synchronous > This function is Non-Reentrant

Table 5-2 Crypto_30_CryWrapper_InitMemory

5.1.3 Crypto_30_CryWrapper_GetVersionInfo

Prototype	
void Crypto_30_CryWrapper_GetVersionInfo (Std_VersionInfoType *versioninfo)	
Parameter	
versioninfo [out]	Pointer to where to store the version information. Parameter must not be NULL.
Return code	
void	none
Functional Description	
Returns the version information.	
Particularities and Limitations	
none Crypto_30_CryWrapper_GetVersionInfo() returns version information, vendor ID and AUTOSAR module ID of the component.	
Call context	
<ul style="list-style-type: none">> TASK ISR> This function is Synchronous> This function is Reentrant	

Table 5-3 Crypto_30_CryWrapper_GetVersionInfo

5.1.4 Crypto_30_CryWrapper_ProcessJob

Prototype	
Std_ReturnType Crypto_30_CryWrapper_ProcessJob (uint32 objectId, Crypto_JobType *job)	
Parameter	
objectId [in]	Holds the identifier of the Crypto Driver Object.
job [in,out]	Pointer to the configuration of the job. Contains structures with job and primitive relevant information but also pointer to result buffers.

Return code	
Std_ReturnType	E_OK Request successful.
	E_NOT_OK Request failed.
	CRYPTO_E_BUSY Request failed, Crypto Driver Object is busy.
	CRYPTO_E_KEY_NOT_VALID Request failed, the key is not valid.
	CRYPTO_E_QUEUE_FULL Request failed, the queue is full.
	CRYPTO_E_SMALL_BUFFER Request failed, the provided buffer is too small to store the result.
Functional Description	
Process the received job.	
Particularities and Limitations	
none	
Performs the crypto primitive that is configured in the job parameter.	
Call context	
<ul style="list-style-type: none">> TASK> This function is Reentrant	

Table 5-4 Crypto_30_CryWrapper_ProcessJob

5.1.5 Crypto_30_CryWrapper_CancelJob

Prototype	
Std_ReturnType Crypto_30_CryWrapper_CancelJob (uint32 objectId, Crypto_JobType *job)	
Parameter	
objectId [in]	Holds the identifier of the Crypto Driver Object.
job [in,out]	Pointer to the configuration of the job. Contains structures with user and primitive relevant information.
Return code	
Std_ReturnType	E_OK Request successful, job has been removed.
Std_ReturnType	E_NOT_OK Request failed, job could not be removed.
Functional Description	
Cancels the received job.	
Particularities and Limitations	
none	
This interface removes the provided job from the queue and cancels the processing of the job if possible.	
Call context	
<ul style="list-style-type: none">> TASK> This function is Synchronous> This function is Reentrant	

Table 5-5 Crypto_30_CryWrapper_CancelJob

5.2 Key Management Functions

5.2.1 Crypto_30_CryWrapper_KeyCopy

Prototype	
Std_ReturnType Crypto_30_CryWrapper_KeyCopy (uint32 cryptoKeyId, uint32 targetCryptoKeyId)	
Parameter	
cryptoKeyId [in]	Holds the identifier of the key whose key element shall be the source element.
targetCryptoKeyId [in]	Holds the identifier of the key whose key element shall be the destination element.
Return code	
Std_ReturnType	E_OK Request successful.
	E_NOT_OK Request failed.
	CRYPTO_E_BUSY Request failed, Crypto Driver Object is busy.
	CRYPTO_E_KEY_READ_FAIL Request failed, read access was denied.
	CRYPTO_E_KEY_WRITE_FAIL Request failed, write access was denied.
	CRYPTO_E_KEY_NOT_AVAILABLE Request failed, the key is not available.
	CRYPTO_E_KEY_SIZE_MISMATCH Request failed, the key element sizes are not compatible.
Functional Description	
Copy the key.	
Particularities and Limitations	
none	
Copies a key with all its elements to another key in the same crypto driver.	
Call context	
<ul style="list-style-type: none"> > TASK > This function is Synchronous > This function is Reentrant 	

Table 5-6 Crypto_30_CryWrapper_KeyCopy

5.2.2 Crypto_30_CryWrapper_KeyElementCopy

Prototype	
Std_ReturnType Crypto_30_CryWrapper_KeyElementCopy (uint32 cryptoKeyId, uint32 keyElementId, uint32 targetCryptoKeyId, uint32 targetKeyElementId)	
Parameter	
cryptoKeyId [in]	Holds the identifier of the key whose key element shall be the source element.

keyElementId [in]	Holds the identifier of the key element which shall be the source for the copy operation.
targetCryptoKeyId [in]	Holds the identifier of the key whose key element shall be the destination element.
targetKeyElementId [in]	Holds the identifier of the key element which shall be the destination for the copy operation.
Return code	
Std_ReturnType	E_OK Request successful.
	E_NOT_OK Request failed.
	CRYPTO_E_BUSY Request failed, Crypto Driver Object is busy.
	CRYPTO_E_KEY_READ_FAIL Request failed, read access was denied.
	CRYPTO_E_KEY_WRITE_FAIL Request failed, write access was denied.
	CRYPTO_E_KEY_EXTRACT_DENIED Request failed, not allowed to extract key material.
	CRYPTO_E_KEY_NOT_AVAILABLE Request failed, the key is not available.
	CRYPTO_E_KEY_SIZE_MISMATCH Request failed, the key element sizes are not compatible.
Functional Description	
Copy key element.	
Particularities and Limitations	
none	
Copies a key element to another key element in the same crypto driver.	
Call context	
<ul style="list-style-type: none"> > TASK > This function is Synchronous > This function is Reentrant 	

Table 5-7 Crypto_30_CryWrapper_KeyElementCopy

5.2.3 Crypto_30_CryWrapper_KeyElementIdsGet

Prototype	
Std_ReturnType Crypto_30_CryWrapper_KeyElementIdsGet (uint32 cryptoKeyId, uint32 *keyElementIdsPtr, uint32 *keyElementIdsLengthPtr)	
Parameter	
cryptoKeyId [in]	Holds the identifier of the key whose available element ids shall be exported.
keyElementIdsLengthPtr [in]	Holds a pointer to the memory location in which the number of key element in the given key is stored. On calling this function, this parameter shall contain the size of the buffer provided by keyElementIdsPtr. When the request has finished, the actual number of key elements is stored.
keyElementIdsPtr [out]	Contains the pointer to the array where the ids of the key elements shall be stored.

Return code	
Std_ReturnType	E_OK Request successful.
	E_NOT_OK Request failed.
	CRYPTO_E_BUSY Request failed, Crypto Driver Object is busy.
	CRYPTO_E_SMALL_BUFFER Request failed, the provided buffer is too small to store the result.
Functional Description	
Used to retrieve information which key elements are available in a given key.	
Particularities and Limitations	
none	
-	
Call context	
<ul style="list-style-type: none">> TASK> This function is Synchronous> This function is Reentrant	

Table 5-8 Crypto_30_CryWrapper_KeyElementIdsGet

5.2.4 Crypto_30_CryWrapper_KeyElementSet

Prototype	
Std_ReturnType Crypto_30_CryWrapper_KeyElementSet (uint32 cryptoKeyId, uint32 keyElementId, const uint8 *keyPtr, uint32 keyLength)	
Parameter	
cryptoKeyId [in]	Holds the identifier of the key whose key element shall be set.
keyElementId [in]	Holds the identifier of the key element which shall be set.
keyPtr [in]	Holds the pointer to the key data which shall be set as key element.
keyLength [in]	Contains the length of the key element in bytes.
Return code	
Std_ReturnType	E_OK Request successful.
	E_NOT_OK Request failed.
	CRYPTO_E_BUSY Request failed, Crypto Driver Object is busy.
	CRYPTO_E_KEY_WRITE_FAIL Request failed, write access was denied.
	CRYPTO_E_KEY_NOT_AVAILABLE Request failed, the key is not available.
	CRYPTO_E_KEY_SIZE_MISMATCH Request failed, the key element size does not match size of provided data.
Functional Description	
Sets a key element.	

Particularities and Limitations
none
Sets the given key element bytes to the key identified by cryptoKeyId. .
Call context
<ul style="list-style-type: none"> > TASK > This function is Synchronous > This function is Reentrant

Table 5-9 Crypto_30_CryWrapper_KeyElementSet

5.2.5 Crypto_30_CryWrapper_KeyValidSet

Prototype	
Std_ReturnType Crypto_30_CryWrapper_KeyValidSet (uint32 cryptoKeyId)	
Parameter	
cryptoKeyId [in]	Holds the identifier of the key whose key elements shall be set to valid.
Return code	
Std_ReturnType	E_OK Request successful.
	E_NOT_OK Request failed.
	CRYPTO_E_BUSY Request failed, Crypto Driver Object is busy.
Functional Description	
Sets the key to valid.	
Particularities and Limitations	
none	
Sets the key state of the key identified by cryptoKeyId to valid.	
Call context	
<div>> TASK</div> <div>> This function is Synchronous</div> <div>> This function is Reentrant</div>	

Table 5-10 Crypto_30_CryWrapper_KeyValidSet

5.2.6 Crypto_30_CryWrapper_KeyElementGet

Prototype	
Std_ReturnType Crypto_30_CryWrapper_KeyElementGet (uint32 cryptoKeyId, uint32 keyElementId, uint8 *resultPtr, uint32 *resultLengthPtr)	
Parameter	
cryptoKeyId [in]	Holds the identifier of the key whose key element shall be set.
keyElementId [in]	Holds the identifier of the key element which shall be set.

resultPtr [in]	Holds the pointer to the key data which shall be set as key element.
resultLengthPtr [in]	Contains the length of the key element in bytes.
Return code	
Std_ReturnType	E_OK Request successful.
	E_NOT_OK Request failed.
	CRYPTO_E_BUSY Request failed, Crypto Driver Object is busy.
	CRYPTO_E_KEY_READ_FAIL Request failed, read access was denied.
	CRYPTO_E_KEY_NOT_AVAILABLE Request failed, the key is not available.
	CRYPTO_E_KEY_SIZE_MISMATCH Request failed, the provided buffer is too small to store the result.
Functional Description	
This interface shall be used to get a key element of the key identified by the cryptoKeyId and store the key element in the memory location pointed by the result pointer.	
Particularities and Limitations	
none	
-	
Call context	
<ul style="list-style-type: none"> > TASK > This function is Synchronous > This function is Reentrant 	

Table 5-11 Crypto_30_CryWrapper_KeyElementGet

5.2.7 Crypto_30_CryWrapper_RandomSeed

Prototype	
Std_ReturnType Crypto_30_CryWrapper_RandomSeed (uint32 cryptoKeyId, const uint8 *entropyPtr, uint32 entropyLength)	
Parameter	
cryptoKeyId [in]	Holds the identifier of the key for which a new seed shall be generated.
entropyPtr [in]	Holds a pointer to the memory location which contains the data to feed the entropy.
entropyLength [in]	Contains the length of the entropy in bytes.
Return code	
Std_ReturnType	E_OK Request successful.
	E_NOT_OK Request failed.
	CRYPTO_E_BUSY Request failed, Crypto Driver Object is busy.
	CRYPTO_E_SMALL_BUFFER Request failed, the provided buffer is too small to store the result.

Functional Description
Initialize the seed.
Particularities and Limitations
none This function generates the internal seed state using the provided entropy source. Furthermore, this function can be used to update the seed state with new entropy.
Call context
<ul style="list-style-type: none">> TASK> This function is Synchronous> This function is Reentrant

Table 5-12 Crypto_30_CryWrapper_RandomSeed

5.2.8 Crypto_30_CryWrapper_KeyGenerate


Prototype	
Std_ReturnType Crypto_30_CryWrapper_KeyGenerate (uint32 cryptoKeyId)	
Parameter	
cryptoKeyId [in]	Holds the identifier of the key which is to be updated with the generated value.
Return code	
Std_ReturnType	E_OK Request successful.
	E_NOT_OK Request failed.
	CRYPTO_E_BUSY Request failed, Crypto Driver Object is busy.
Functional Description	
Generates a key.	
Particularities and Limitations	
<div>Note This API does not offer any functionality and is only available for compatibility reasons.</div>	
Call context	
<ul style="list-style-type: none">> TASK> This function is Synchronous> This function is Reentrant	

Table 5-13 Crypto_30_CryWrapper_KeyGenerate

5.2.9 Crypto_30_CryWrapper_KeyDerive


Prototype	
Std_ReturnType Crypto_30_CryWrapper_KeyDerive (uint32 cryptoKeyId, uint32 targetCryptoKeyId)	
Parameter	
cryptoKeyId [in]	Holds the identifier of the key which is used for key derivation.
targetCryptoKeyId [in]	Holds the identifier of the key which is used to store the derived key.
Return code	
Std_ReturnType	E_OK Request successful.
	E_NOT_OK Request failed.
	CRYPTO_E_BUSY Request failed, Crypto Driver Object is busy.
Functional Description	
Derives a key.	
Particularities and Limitations	
<div>Note This API does not offer any functionality and is only available for compatibility reasons.</div>	
Call context	
<ul style="list-style-type: none">> TASK> This function is Synchronous> This function is Reentrant	

Table 5-14 Crypto_30_CryWrapper_KeyDerive

5.2.10 Crypto_30_CryWrapper_KeyExchangeCalcPubVal

Prototype	
Std_ReturnType Crypto_30_CryWrapper_KeyExchangeCalcPubVal (uint32 cryptoKeyId, uint8 *publicValuePtr, uint32 *publicValueLengthPtr)	
Parameter	
cryptoKeyId [in]	Holds the identifier of the key which shall be used for the key exchange protocol.
publicValuePtr [out]	Contains the pointer to the data where the public value shall be stored.
publicValueLengthPtr [in,out]	Holds a pointer to the memory location in which the public value length information is stored. On calling this function, this parameter shall contain the size of the buffer provided by publicValuePtr. When the request has finished, the actual length of the returned value shall be stored.


Return code	
Std_ReturnType	E_OK Request successful.
	E_NOT_OK Request failed.
	CRYPTO_E_BUSY Request failed, Crypto Driver Object is busy.
	CRYPTO_E_SMALL_BUFFER Request failed, the provided buffer is too small to store the result.
Functional Description	
Calculation of the public value.	
Particularities and Limitations	
<div>Note This API does not offer any functionality and is only available for compatibility reasons.</div>	
Call context	
<ul style="list-style-type: none">> TASK> This function is Synchronous> This function is Reentrant	

Table 5-15 Crypto_30_CryWrapper_KeyExchangeCalcPubVal

5.2.11 Crypto_30_CryWrapper_KeyExchangeCalcSecret

Prototype	
Std_ReturnType Crypto_30_CryWrapper_KeyExchangeCalcSecret (uint32 cryptoKeyId, const uint8 *partnerPublicValuePtr, uint32 partnerPublicValueLength)	
Parameter	
cryptoKeyId [in]	Holds the identifier of the key which shall be used for the key exchange protocol.
partnerPublicValuePtr [in]	Holds the pointer to the memory location which contains the partners public value.
partnerPublicValueLength [in]	Contains the length of the partners public value in bytes.
Return code	
Std_ReturnType	E_OK Request successful.
	E_NOT_OK Request failed.
	CRYPTO_E_BUSY Request failed, Crypto Driver Object is busy.
	CRYPTO_E_SMALL_BUFFER Request failed, the provided buffer is too small to store the result.



Functional Description	
Calculation of the secret.	
Particularities and Limitations	
<div><div></div><div>Note This API does not offer any functionality and is only available for compatibility reasons.</div></div>	
Call context	
<ul style="list-style-type: none">> TASK> This function is Synchronous> This function is Reentrant	

Table 5-16 Crypto_30_CryWrapper_KeyExchangeCalcSecret

5.2.12 Crypto_30_CryWrapper_CertificateParse

Prototype	
Std_ReturnType Crypto_30_CryWrapper_CertificateParse (uint32 cryptoKeyId)	
Parameter	
cryptoKeyId [in]	Holds the identifier of the key slot in which the certificate has been stored.
Return code	
Std_ReturnType	E_OK Request successful.
	E_NOT_OK Request failed.
	CRYPTO_E_BUSY Request failed, Crypto Driver Object is busy.
Functional Description	
Parse stored certificate.	
Particularities and Limitations	
<div><div></div><div>Note This API does not offer any functionality and is only available for compatibility reasons.</div></div>	
Call context	
<ul style="list-style-type: none">> TASK> This function is Synchronous	

> This function is Reentrant

Table 5-17 Crypto_30_CryWrapper_CertificateParse

5.2.13 Crypto_30_CryWrapper_CertificateVerify


Prototype	
Std_ReturnType Crypto_30_CryWrapper_CertificateVerify (uint32 cryptoKeyId, uint32 verifyCryptoKeyId, Crypto_VerifyResultType *verifyPtr)	
Parameter	
cryptoKeyId [in]	Holds the identifier of the key which shall be used to validate the certificate.
verifyCryptoKeyId [in]	Holds the identifier of the key containing the certificate, which shall be verified.
verifyPtr [out]	Holds a pointer to the memory location which will contain the result of the certificate verification.
Return code	
Std_ReturnType	E_OK Request successful.
	E_NOT_OK Request failed.
	CRYPTO_E_BUSY Request failed, Crypto Driver Object is busy.
Functional Description	
Certificate verification.	
Particularities and Limitations	
<div>Note This API does not offer any functionality and is only available for compatibility reasons.</div>	
Call context	
<p>> TASK</p> <p>> This function is Synchronous</p> <p>> This function is Reentrant</p>	

Table 5-18 Crypto_30_CryWrapper_CertificateVerify

5.3 Services used by CRYPTO

In the following table services provided by other components, which are used by the CRYPTO are listed. For details about prototype and functionality refer to the documentation of the providing component.

Component	API
DET	Det_ReportError

Table 5-19 Services used by the CRYPTO

6 Configuration

In the CRYPTO the attributes can be configured according to/ with the following methods/ tools:

- > Configuration in DaVinci Configurator 5 Pro

6.1 Configuration Variants

The CRYPTO supports the configuration variants

- > VARIANT-PRE-COMPILE

The configuration classes of the CRYPTO parameters depend on the supported configuration variants. For their definitions please see the Crypto_30_CryWrapper_bswmd.arxml file.

7 Glossary and Abbreviations

7.1 Glossary

Term	Description
CSM	Crypto Service Manager
CRYIF	Crypto Interface
CRYPTO	Crypto Driver

Table 7-1 Glossary

7.2 Abbreviations

Abbreviation	Description
API	Application Programming Interface
AUTOSAR	Automotive Open System Architecture
BSW	Basis Software
DEM	Diagnostic Event Manager
DET	Development Error Tracer
EAD	Embedded Architecture Designer
ECU	Electronic Control Unit
HIS	Hersteller Initiative Software
ISR	Interrupt Service Routine
MICROSAR	Microcontroller Open System Architecture (the Vector AUTOSAR solution)
PPORT	Provide Port
RPORT	Require Port
RTE	Runtime Environment
SRS	Software Requirement Specification
SWC	Software Component
SWS	Software Specification
SHE	Secure Hardware Extension

Table 7-2 Abbreviations

8 Contact

Visit our website for more information on

- > News
- > Products
- > Demo software
- > Support
- > Training data
- > Addresses

www.vector.com