MICROSAR CRY

Technical Reference

Version 2.0

Authors	Philipp Ritter, Markus Schneider, Tobias Finke
Status	Released

Document Information

History

Author	Date	Version	Remarks
Philipp Ritter	2012-10-01	1.00.00	Initial Version of MICROSAR CSM
Markus Schneider	2015-03-24	1.01.00	Added FIPS-186.2 and HMAC SHA-1; adapted configuration chapter
Tobias Finke	2015-04-23	1.02.00	Added RSA-1024 Decrypt and RSA-SHA-1 Signature Verification
Tobias Finke	2015-07-30	1.02.01	Refactoring of key types
Markus Schneider	2015-12-08	2.00.00	Added DaVinci Configurator 5 support

Reference Documents

No.	Source	Title	Version
[1]	AUTOSAR	AUTOSAR_SWS_CryptoServiceManager.pdf	1.2.0
[2]	AUTOSAR	AUTOSAR_SWS_DevelopmentErrorTracer.pdf	3.2.0
[3]	AUTOSAR	AUTOSAR_SWS_DiagnosticEventManager.pdf	4.2.0
[4]	AUTOSAR	AUTOSAR_TR_BSWModuleList.pdf	1.6.0
[5]	AUTOSAR	AUTOSAR_SWS_RTE.pdf	3.2.0

Caution
We have configured the programs in accordance with your specifications in the questionnaire. Whereas the programs do support other configurations than the one specified in your questionnaire, Vector´s release of the programs delivered to your company is expressly restricted to the configuration you have specified in the questionnaire.

Caution
This symbol calls your attention to warnings.

Contents

Illustrations

Tables

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
1.0	Initial version
1.1	Added FIPS-186.2 and HMAC SHA-1
1.2	Added RSA-1024 Decrypt and RSA-SHA-1 Signature Verification
2.0	Support of DaVinci Configurator 5

Table 1-1 Component history

2 Introduction

This document describes the functionality, API and configuration of the MICROSAR module CRY as specified in [1].

Supported AUTOSAR Release*:	4	
Supported Configuration Variants:	pre-compile	
Vendor ID:	CRY_VENDOR_ID	30 decimal (= Vector-Informatik, according to HIS)
Module ID:	CRY_MODULE_ID	255 decimal (according to ref. [4])

* For the precise AUTOSAR Release 4.x please see the release specific documentation.

The Cryptographic library module (CRY) offers cryptographic primitives. The CRY module is used by the Crypto Service Manager (CSM).

2.1 Architecture Overview

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

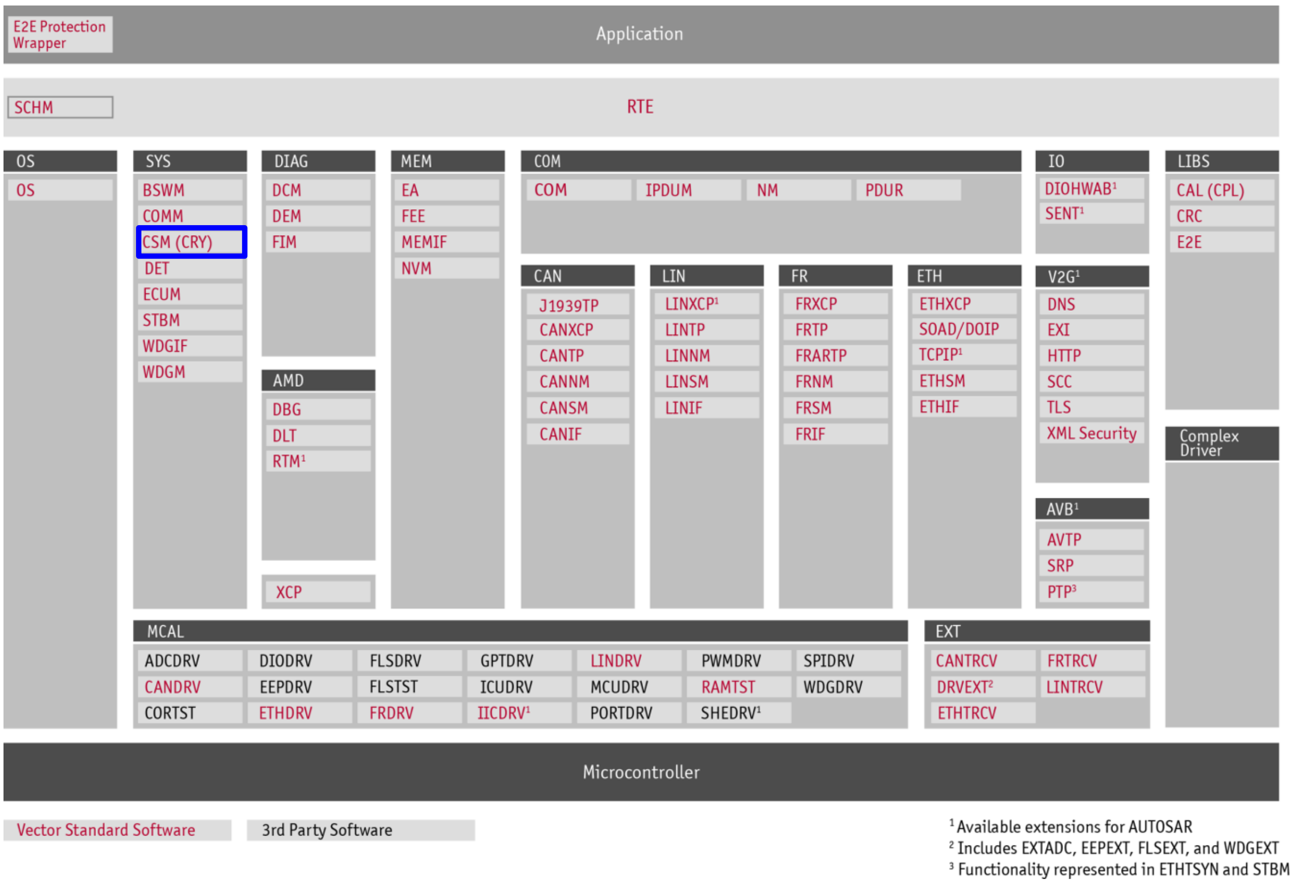

Figure 2-1 AUTOSAR 4.x Architecture Overview

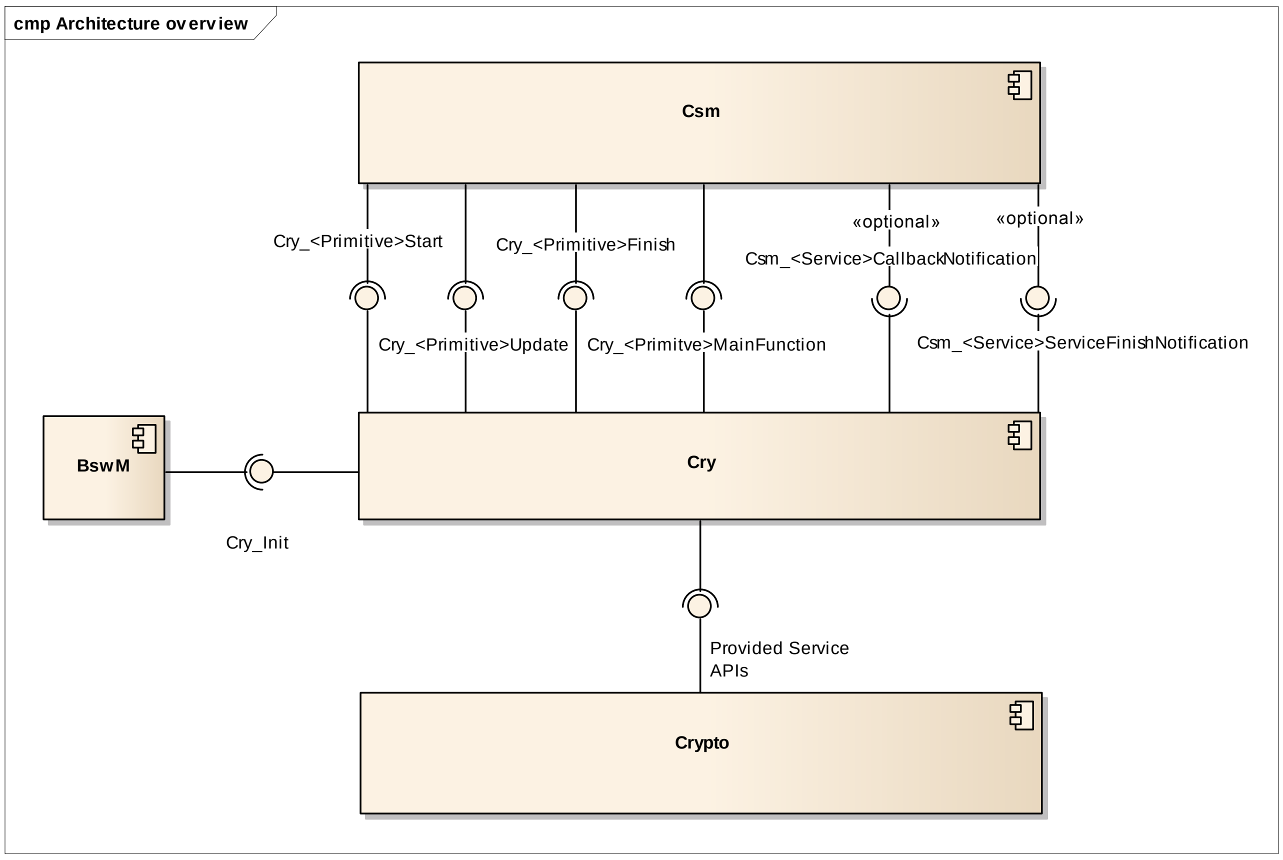

Figure 2-2 Interfaces to adjacent modules of the CRY

3 Functional Description

3.1 Features

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

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

For further information of not supported features see also chapter 7.

The following features specified in [1] are supported:

Supported AUTOSAR Standard Conform Features
Synchronous job processing
Asynchronous job processing
Service for Symmetrical Interface (AES128)
Service for MAC Interface (HMAC SHA-1)
Service for Random Interface (FIPS-186.2)
Service for Asymmetrical Interface
Service for Signature Interface

Table 3-1 Supported AUTOSAR standard conform features

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

Not Supported AUTOSAR Standard Conform Features
Service for Hash Interface
Service for Symmetrical Block Interface
Service for Checksum Interface
Service for Key Derivation Interface
Service for Key Exchange Interface
Service for Symmetrical Key Exchange Interface
Service for Symmetrical Key Extract Interface
Service for Asymmetrical Key Extract Interface

Table 3-2 Not supported AUTOSAR standard conform features

3.2 Initialization

Before calling any other functionality of the Cry module the initialization function `Cry_Init()` has to be called. For API details refer to chapter 5.3.1 ‘Cry_Init’.

3.3 Main Functions

The CRY module implementation provides a main function for each service. When the usage of sync job processing is disabled, this main function has to be called by the CSM whenever a service is active.

For API details refer e.g. to chapter 5.3.7 ‘Cry_AesEncrypt128MainFunction’.

3.4 Key Handling

The asymmetrical keys used by the Cry module are in the format of ‘Cry_RsaKeyType’, which is defined in ‘Cry_Key_Types.h’.

3.5 Error Handling

3.5.1 Development Error Reporting

The current implementation of the Cry module does not report any development errors.

3.5.2 Production Code Error Reporting

The current implementation of the Cry module does not report any production errors.

4 Integration

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

4.1 Scope of Delivery

The delivery of the CRY contains the files which are described in the chapters 4.1.1 and 4.1.2.

4.1.1 Static Files

File Name	Source Code Delivery	Library Delivery	Description
Cry.c	■		Source file of the Cry.
Cry.h	■		Header file of the Cry.
Cry_ AesDecrypt128.c	■		Source file of the service AesDecrypt128.
Cry_ AesDecrypt128.h	■		Header file of the service AesDecrypt128.
Cry_ AesEncrypt128.c	■		Source file of the service AesEncrypt128.
Cry_ AesEncrypt128.h	■		Header file of the service AesEncrypt128.
Cry_ Fips186.c	■		Source file of the service FIPS-186.
Cry_ Fips186.h	■		Header file of the service FIPS-186.
Cry_ HmacSha1.c	■		Source file of the service HMAC SHA-1.
Cry_ HmacSha1.h	■		Header file of the service HMAC SHA-1.
SecModLib.lib[1]		■	Library file of the cryptographic primitives
Cry_RsaDecrypt1024.c	■		Source file of the service RsaDecrypt1024.
Cry_RsaDecrypt1024.h	■		Header file of the service RsaDecrypt1024.
Cry_RsaSha1SigVer.c	■		Source file of the service RsaSha1SigVer.
Cry_RsaSha1SigVer.h	■		Header file of the service RsaSha1SigVer.
Cry_Key_Types.h	■		Header file for custom key types

Table 4-1 Static files

[1] The name of the underlying cryptographic primitive library may differ.

4.1.2 Dynamic Files

The dynamic files are generated by the DaVinci Configurator 5.

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

Table 4-2 Generated files

4.2 Include Structure

Figure 4-1 shows the include structure of the Cry. Some includes are optional and depend on the configuration. `Cry_<Primitve>.h` stands for every used cryptographic primitive.

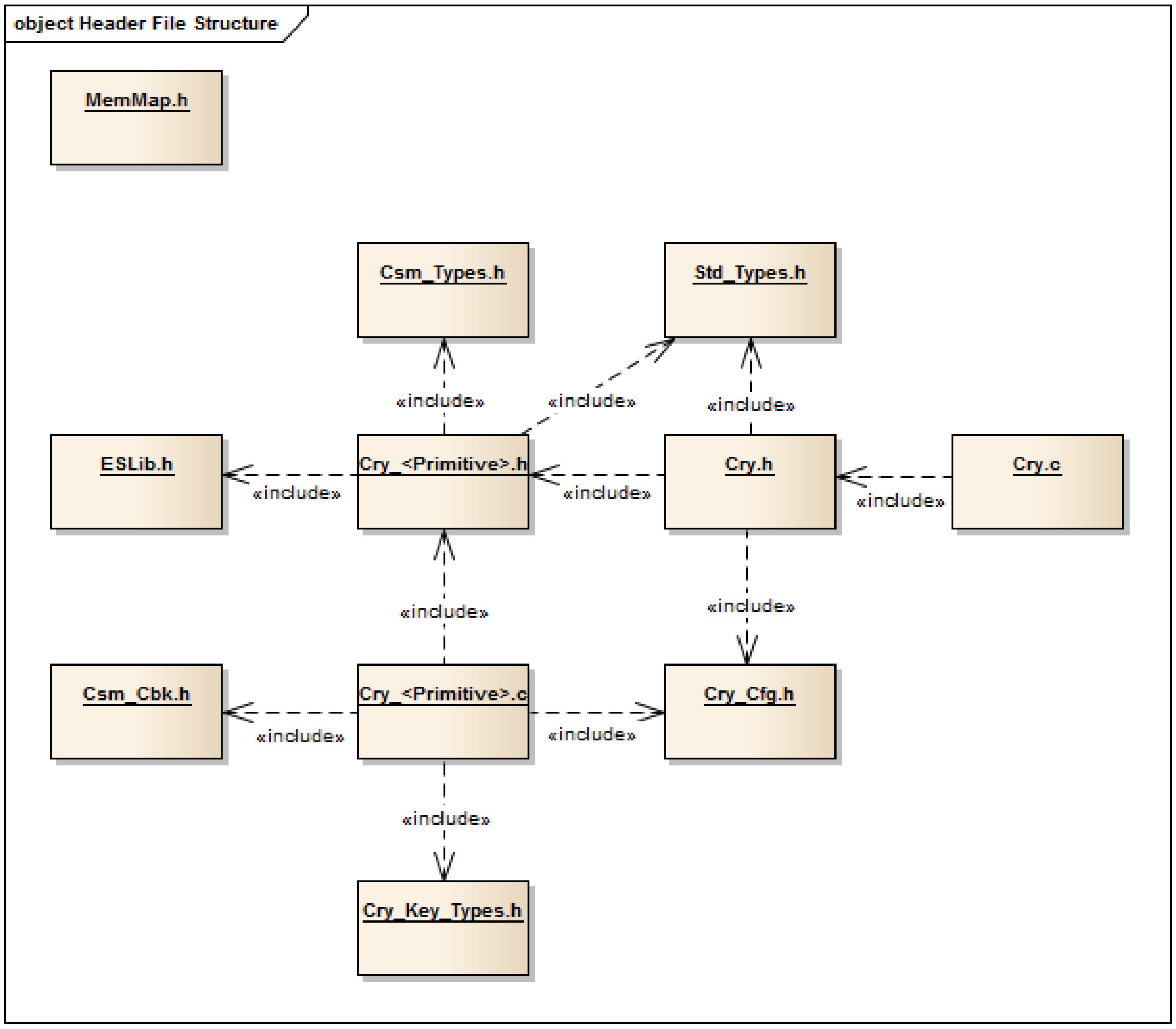

Figure 4-1 Include structure

4.3 Compiler Abstraction and Memory Mapping

The objects (e.g. variables, functions, constants) are declared by compiler independent definitions – the compiler abstraction definitions. Each compiler abstraction definition is assigned to a memory section.

The following table (Table 4-3) contains the memory section names and the compiler abstraction definitions of the CRY and illustrates their assignment among each other.

Compiler Abstraction Definitions / Memory Mapping Sections	CRY_CODE	CRY_VAR_NOINIT	CRY_APPL_VAR
CRY_START_SEC_CODE CRY_STOP_SEC_CODE	■		■
CRY_START_SEC_VAR_NOINIT_8BIT CRY_STOP_SEC_VAR_NOINIT_8BIT		■	
CRY_START_SEC_VAR_NOINIT_UNSPECIFIED CRY_STOP_SEC_VAR_NOINIT_UNSPECIFIED		■	

Table 4-3 Compiler abstraction and memory mapping

4.4 Critical Sections

The current implementation of the CRY module does not have any critical section.

5 API Description

5.1 Interfaces Overview

For an interfaces overview please see Figure 2-2.

5.2 Structures

5.2.1 Cry_Aes128ConfigType

This structure represents the configuration for the AesDecrypt128 and AesEncrypt128 service

Struct Element Name	C-Type	Description	Value Range
buffer	Cry_Aes128WorkSpaceType*	Pointer to a provided buffer which will be used as workspace for the primitives	
blockMode	uint8	Block mode	`CRY_BLOCKMODE_ECB`, `CRY_BLOCKMODE_CBC`
paddingMode	uint8	Padding mode	`CRY_PADDINGMODE_PKCS5`

Table 5-1 Cry_Aes128ConfigType

5.2.2 Cry_Fips186ConfigType

This structure represents the configuration for the FIPS-186 service

Struct Element Name	C-Type	Description	Value Range
buffer	Cry_Fips186WorkSpaceType*	Pointer to a provided buffer which will be used as workspace for the primitives	
savedStateEnabled	boolean	Enable storing the RNG state. This shall always be `TRUE`. Except for debugging or testing reasons.	`TRUE`, `FALSE`

Table 5-2 Cry_ Fips186ConfigType

5.2.3 Cry_HmacSha1ConfigType

This structure represents the configuration for the HMAC SHA-1 service

Struct Element Name	C-Type	Description	Value Range
buffer	Cry_HmacSha1WorkSpaceType *	Pointer to a provided buffer which will be used as workspace for the primitives	

Table 5-3 Cry_ HmacSha1ConfigType

5.2.4 Cry_RsaDecrypt1024ConfigType

This structure represents the configuration for the RsaDecrypt1024 service

Struct Element Name	C-Type	Description	Value Range
buffer	Cry_RsaDecrypt1024WorkSpaceType *	Pointer to a provided buffer which will be used as workspace for the primitives	

Table 5-4 Cry_ RsaDecrypt1024ConfigType

5.2.5 Cry_RsaSha1SigVerConfigType

This structure represents the configuration for the RsaSha1SigVer service

Struct Element Name	C-Type	Description	Value Range
buffer	Cry_RsaSha1SigVerWorkSpaceType *	Pointer to a provided buffer which will be used as workspace for the primitives	

Table 5-5 Cry_ RsaSha1SigVerConfigType

5.2.6 Cry_RsaKeyType

This structure represents a 1024 Bit RSA key which is defined by a modulo and an exponent. This structure is used for passing a key to the RsaDecrypt1024 and RsaSha1SigVer service.

Struct Element Name	C-Type	Description	Value Range
keyModuleLength	uint16	Length of the modulo in byte.	
keyModule	const uint8 *	Pointer to the modulo of the key.	
keyExponentLength	uint16	Length of the exponent in byte.	
keyExponent	const uint8 *	Pointer to the exponent of the key.	

Table 5-6 Cry_ RsaKeyType

5.3 Services provided by CRY

5.3.1 Cry_Init

Prototype	
void **Cry_Init** (void)	
Parameter	
-	
Return code	
-	
Functional Description	
This function initializes the Cry.	
Particularities and Limitations	
> This function is synchronous. > This function is non-reentrant. > This function has to be called during start-up.	
Call Context	
> This function can be called from task level only.	

Table 5-7 Cry_Init

5.3.2 Cry_InitMemory

Prototype	
`void` **`Cry_InitMemory`** `(void)`	
Parameter	
-	
Return code	
-	
Functional Description	
This function is currently empty but required by the MICROSAR stack.	
Particularities and Limitations	
> This function is synchronous. > This function is non-reentrant.	
Call Context	
> This function can be called from task level only.	

Table 5-8 Cry_ InitMemory

5.3.3 Cry_GetVersionInfo

Prototype	
`void` **`Cry_GetVersionInfo`** `(Std_VersionInfoType *cryVerInfoPtr)`	
Parameter	
`cryVerInfoPtr`	Pointer where the version information shall be copied to.
Return code	
-	
Functional Description	
This function copies the Cry version information to the location provided by the pointer.	
Particularities and Limitations	
> This function is synchronous. > This function is non-reentrant. > This function is only available if 'Version Info Api" is enabled.	
Call Context	
> This function can be called from task and interrupt level.	

Table 5-9 Cry_GetVersionInfo

5.3.4 Cry_AesEncrypt128Start

Prototype	
`Csm_ReturnType` **`Cry_AesEncrypt128Start`** `(Const void *cfgPtr, const Csm_SymKeyType *keyPtr, const uint8 *InitVectorPtr, uint32 InitVectorLength)`	
Parameter	
cfgPtr	Holds a pointer to the configuration of this service. See Cry_Aes128ConfigType for more information.
keyPtr	Holds a pointer to the key which has to be used during the symmetrical encryption operation.
InitVectorPtr	Holds a pointer to initialization vector which has to be used during the symmetrical encryption.
InitVectorLength	Holds a pointer to the initialization vector which has to be used during the symmetrical encryption.
Return code	
CSM_E_OK	Request successful.
CSM_E_NOT_OK	Request failed.
Functional Description	
This interface shall be used to initialize the symmetrical encryption service.	
Particularities and Limitations	
> This function can be synchronous or asynchronous. > This function is non-reentrant. > This function is called by application.	
Call Context	
> This function can be called from task level only.	

Table 5-10 Cry_AesEncrypt128Start

5.3.5 Cry_AesEncrypt128Update

Prototype	
`Csm_ReturnType Cry_AesEncrypt128Update (Const void *cfgPtr, const uint8 *plainTextPtr, uint32 plainTextLength, uint8 *cipherTextPtr, uint32 *cipherTextLengthPtr)`	
Parameter	
cfgPtr	Holds a pointer to the configuration of this service. See Cry_Aes128ConfigType for more information.
plainTextPtr	Holds a pointer to the data for which a encrypted text shall be computed.
plainTextLength	Contains the number of bytes for which the encrypted text shall be computed.
cipherTextPtr	Holds a pointer to the memory location which will hold the encrypted text.
cipherTextLengthPtr	Holds a pointer to the memory location in which the length information is stored. On calling this function this parameter shall contain the size of the provided buffer. When the request has finished, the actual length of the returned encrypted text shall be stored.
Return code	
CSM_E_OK	Request successful.
CSM_E_NOT_OK	Request failed.
CSM_E_SMALL_BUFFER	The provided buffer is too small to store the result and truncation was not allowed.
Functional Description	
This interface shall be used to feed the symmetrical encryption service with the input data.	
Particularities and Limitations	
> This function can be synchronous or asynchronous. > This function is non-reentrant. > This function is called by application.	
Call Context	
> This function can be called from task level only.	

Table 5-11 Cry_AesEncrypt128Update

5.3.6 Cry_AesEncrypt128Finish

Prototype	
`Csm_ReturnType` **`Cry_AesEncrypt128Finish`** `(Const void *cfgPtr, uint8 *cipherTextPtr, uint32 *cipherTextLengthPtr)`	
Parameter	
cfgPtr	Holds a pointer to the configuration of this service. See Cry_Aes128ConfigType for more information.
cipherTextPtr	Holds a pointer to the memory location which will hold the encrypted text.
cipherTextLengthPtr	Holds a pointer to the memory location in which the length information is stored. On calling this function this parameter shall contain the size of the provided buffer. When the request has finished, the actual length of the returned encrypted text shall be stored.
Return code	
CSM_E_OK	Request successful.
CSM_E_NOT_OK	Request failed.
CSM_E_SMALL_BUFFER	The provided buffer is too small to store the result and truncation was not allowed.
Functional Description	
This interface shall be used to finish the symmetrical encryption service.	
Particularities and Limitations	
> This function can be synchronous or asynchronous. > This function is non-reentrant. > This function is called by application.	
Call Context	
> This function can be called from task level only.	

Table 5-12 Cry_AesEncrypt128Finish

5.3.7 Cry_AesEncrypt128MainFunction

Prototype	
`void` **`Cry_AesEncrypt128MainFunction`** `(void)`	
Parameter	
-	
Return code	
-	
Functional Description	
This function implements the asynchronous service handling. **Note** This function is empty if 'Use Sync Job Processing' is enabled.	
Particularities and Limitations	
> This function is synchronous. > This function is not reentrant. > This function has to be called by CSM. > This function must not be called by the application.	
Call Context	
> This function can be called from task level only.	

Table 5-13 Cry_AesEncrypt128MainFunction

5.3.8 Cry_AesDecrypt128Start

Prototype	
`Csm_ReturnType` **`Cry_AesDecrypt128Start`** `(Const void *cfgPtr, const Csm_SymKeyType *keyPtr, const uint8 *InitVectorPtr, uint32 InitVectorLength)`	
Parameter	
cfgPtr	Holds a pointer to the configuration of this service. See Cry_Aes128ConfigType for more information.
keyPtr	Holds a pointer to the key which has to be used during the symmetrical decryption operation.
InitVectorPtr	Holds a pointer to initialization vector which has to be used during the symmetrical decryption.
InitVectorLength	Holds a pointer to the initialization vector which has to be used during the symmetrical decryption.
Return code	
CSM_E_OK	Request successful.
CSM_E_NOT_OK	Request failed.
Functional Description	
This interface shall be used to initialize the symmetrical decryption service of the CSM module.	
Particularities and Limitations	
> This function can be synchronous or asynchronous. > This function is non-reentrant. > This function is called by application.	
Call Context	
> This function can be called from task level only.	

Table 5-14 Cry_AesDecrypt128Start

5.3.9 Cry_AesDecrypt128Update

Prototype	
`Csm_ReturnType` **`Cry_AesDecrypt128Update`** `(Const void *cfgPtr, const uint8 *cipherTextPtr, uint32 cipherTextLength, uint8 *plainTextPtr, uint32 *plainTextLengthPtr)`	
Parameter	
cfgPtr	Holds a pointer to the configuration of this service. See Cry_Aes128ConfigType for more information.
cipherTextPtr	Holds a pointer to the data for which a decrypted text shall be computed.
cipherTextLength	Contains the number of bytes for which the decrypted text shall be computed.
plainTextPtr	Holds a pointer to the memory location which will hold the decrypted text.
plainTextLengthPtr	Holds a pointer to the memory location in which the length information is stored. On calling this function this parameter shall contain the size of the provided buffer. When the request has finished, the actual length of the returned decrypted text shall be stored.
Return code	
CSM_E_OK	Request successful.
CSM_E_NOT_OK	Request failed.
CSM_E_SMALL_BUFFER	The provided buffer is too small to store the result and truncation was not allowed.
Functional Description	
This interface shall be used to feed the symmetrical decryption service with the input data.	
Particularities and Limitations	
> This function can be synchronous or asynchronous. > This function is non-reentrant. > This function is called by application.	
Call Context	
> This function can be called from task level only.	

Table 5-15 Cry_AesDecrypt128Update

5.3.10 Cry_AesDecrypt128Finish

Prototype	
`Csm_ReturnType` **`Cry_AesDecrypt128Finish`** `(Const void *cfgPtr, uint8 *plainTextPtr, uint32 *plainTextLengthPtr)`	
Parameter	
cfgPtr	Holds a pointer to the configuration of this service. See Cry_Aes128ConfigType for more information.
plainTextPtr	Holds a pointer to the memory location which will hold the decrypted text.
plainTextLengthPtr	Holds a pointer to the memory location in which the length information is stored. On calling this function this parameter shall contain the size of the provided buffer. When the request has finished, the actual length of the returned decrypted text shall be stored.
Return code	
CSM_E_OK	Request successful.
CSM_E_NOT_OK	Request failed.
CSM_E_SMALL_BUFFER	The provided buffer is too small to store the result and truncation was not allowed.
Functional Description	
This interface shall be used to finish the symmetrical decryption service.	
Particularities and Limitations	
> This function can be synchronous or asynchronous. > This function is non-reentrant. > This function is called by application.	
Call Context	
> This function can be called from task level only.	

Table 5-16 Cry_AesDecrypt128Finish

5.3.11 Cry_AesDecrypt128MainFunction

Prototype	
`void` **`Cry_AesDecrypt128MainFunction`** `(void)`	
Parameter	
-	
Return code	
-	
Functional Description	
This function implements the asynchronous service handling. **Note** This function is empty if 'Use Sync Job Processing' is enabled.	
Particularities and Limitations	
> This function is synchronous. > This function is not reentrant. > This function has to be called by CSM. > This function must not be called by the application.	
Call Context	
> This function can be called from task level only.	

Table 5-17 Cry_AesDecrypt128MainFunction

5.3.12 Cry_Fips186SeedStart

Prototype	
`Csm_ReturnType` **`Cry_ Fips186SeedStart`** `(Const void *cfgPtr)`	
Parameter	
cfgPtr	Holds a pointer to the configuration of this service. See Cry_Fips186ConfigType for more information.
Return code	
CSM_E_OK	Request successful.
CSM_E_NOT_OK	Request failed.
Functional Description	
This function initializes the workspace for the random seed service.	
Particularities and Limitations	
> This function can be synchronous or asynchronous. > This function is non-reentrant. > This function is called by application.	

Call Context
> This function can be called from task level only.

Table 5-18 Cry_ Fips186SeedStart

5.3.13 Cry_Fips186SeedUpdate

Prototype	
`Csm_ReturnType` **`Cry_ Fips186SeedUpdate`** `(Const void *cfgPtr, const uint8 *seedPtr, uint32 seedLength)`	
Parameter	
cfgPtr	Holds a pointer to the configuration of this service. See Cry_Fips186ConfigType for more information.
seedPtr	Holds a pointer to the seed for the random number generator.
seedLength	Contains the length of the seed in bytes.
Return code	
CSM_E_OK	Request successful.
CSM_E_NOT_OK	Request failed.
CSM_E_SMALL_BUFFER	The provided buffer is too small to store the result and truncation was not allowed.
Functional Description	
This function shall be used to feed a seed to the random number generator.	
Particularities and Limitations	
> This function can be synchronous or asynchronous. > This function is non-reentrant. > This function is called by application.	
Call Context	
> This function can be called from task level only.	

Table 5-19 Cry_ Fips186SeedUpdate

5.3.14 Cry_Fips186SeedFinish

Prototype	
`Csm_ReturnType **Cry_ Fips186SeedFinish** (Const void *cfgPtr)`	
Parameter	
cfgPtr	Holds a pointer to the configuration of this service. See Cry_Fips186ConfigType for more information.
Return code	
CSM_E_OK	Request successful.
Functional Description	
This function finalizes the random seed service.	
Particularities and Limitations	
> This function can be synchronous or asynchronous. > This function is non-reentrant. > This function is called by application.	
Call Context	
> This function can be called from task level only.	

Table 5-20 Cry_ Fips186SeedFinish

5.3.15 Cry_Fips186SeedMainFunction

Prototype	
`void **Cry_ Fips186SeedMainFunction** (void)`	
Parameter	
-	
Return code	
-	
Functional Description	
This function implements the asynchronous service handling. **Note** This function is empty if ‘Use Sync Job Processing’ is enabled.	
Particularities and Limitations	
> This function is synchronous. > This function is not reentrant. > This function has to be called by CSM. > This function must not be called by the application.	
Call Context	
> This function can be called from task level only.	

Table 5-21 Cry_ Fips186SeedMainFunction

5.3.16 Cry_Fips186Generate

Prototype	
`Csm_ReturnType` **`Cry_ Fips186Generate`** `(Const void *cfgPtr, uint8 *resultPtr, uint32 resultLength)`	
Parameter	
cfgPtr	Holds a pointer to the configuration of this service. See Cry_Fips186ConfigType for more information.
resultPtr	Holds a pointer to the memory location which will hold the result of the random number generation. The memory location must have at least the size "resultLength".
resultLength	Holds the amount of random bytes which should be generated.
Return code	
CSM_E_OK	Request successful.
CSM_E_NOT_OK	Request failed.
Functional Description	
Generates a random number according to the FIPS186-2 specification.	
Particularities and Limitations	
> This function can be synchronous or asynchronous. > This function is non-reentrant. > This function is called by application.	
Call Context	
> This function can be called from task level only.	

Table 5-22 Cry_ Fips186Generate

5.3.17 Cry_Fips186GenerateMainFunction

Prototype	
`void Cry_ Fips186GenerateMainFunction (void)`	
Parameter	
-	
Return code	
-	
Functional Description	
This function implements the asynchronous service handling. **Note** This function is empty if ‘Use Sync Job Processing’ is enabled.	
Particularities and Limitations	
> This function is synchronous. > This function is not reentrant. > This function has to be called by CSM. > This function must not be called by the application.	
Call Context	
> This function can be called from task level only.	

Table 5-23 Cry_ Fips186GenerateMainFunction

5.3.18 Cry_HmacSha1VerifyStart

Prototype	
`Csm_ReturnType **Cry_ HmacSha1VerifyStart** (Const void *cfgPtr, const Csm_SymKeyType *keyPtr)`	
Parameter	
cfgPtr	Holds a pointer to the configuration of this service. See Cry_HmacSha1ConfigType for more information.
keyPtr	Holds a pointer to the key.
Return code	
CSM_E_OK CSM_E_NOT_OK	Request successful. Request failed.
Functional Description	
This interface shall be used to initialize the HMAC SHA1 verification.	
Particularities and Limitations	
> This function can be synchronous or asynchronous. > This function is non-reentrant. > This function is called by application.	
Call Context	
> This function can be called from task level only.	

Table 5-24 Cry_ HmacSha1VerifyStart

5.3.19 Cry_HmacSha1VerifyUpdate

Prototype	
`Csm_ReturnType` **`Cry_ HmacSha1VerifyUpdate`** `(Const void *cfgPtr, const uint8 *dataPtr, uint32 dataLength)`	
Parameter	
cfgPtr	Holds a pointer to the configuration of this service. See Cry_HmacSha1ConfigType for more information.
dataPtr	Holds a pointer to the seed for the random number generator.
dataLength	Contains the length of the seed in bytes.
Return code	
CSM_E_OK CSM_E_NOT_OK CSM_E_SMALL_BUFFER	Request successful. Request failed. The provided buffer is too small to store the result and truncation was not allowed.
Functional Description	
This function shall be used to feed a seed to the random number generator.	
Particularities and Limitations	
> This function can be synchronous or asynchronous. > This function is non-reentrant. > This function is called by application.	
Call Context	
> This function can be called from task level only.	

Table 5-25 Cry_ HmacSha1VerifyUpdate

5.3.20 Cry_HmacSha1VerifyFinish

Prototype	
`Csm_ReturnType` **`Cry_ HmacSha1VerifyFinish`** `(Const void *cfgPtr, const uint8 *MacPtr, uint32 MacLength, Csm_VerifyResultType *resultPtr)`	
Parameter	
cfgPtr	Holds a pointer to the configuration of this service. See Cry_HmacSha1ConfigType for more information.
MacPtr	Holds a pointer to the memory location which will hold the MAC to verify.
MacLength	Holds the length of the MAC to be verified.
resultPtr	Holds a pointer to the memory location which will hold the MAC.
Return code	
CSM_E_OK	Request successful.
CSM_E_NOT_OK	Request failed
CSM_E_BUSY	Request failed, service is busy
Functional Description	
This interface shall be used to finish the HMAC SHA1 verification.	
Particularities and Limitations	
> This function can be synchronous or asynchronous. > This function is non-reentrant. > This function is called by application.	
Call Context	
> This function can be called from task level only.	

Table 5-26 Cry_ HmacSha1VerifyFinish

5.3.21 Cry_HmacSha1VerifyMainFunction

Prototype	
`void` **`Cry_ HmacSha1VerifyMainFunction`** `(void)`	
Parameter	
-	
Return code	
-	
Functional Description	
This function implements the asynchronous service handling. **Note** This function is empty if 'Use Sync Job Processing' is enabled.	
Particularities and Limitations	
> This function is synchronous. > This function is not reentrant. > This function has to be called by CSM. > This function must not be called by the application.	
Call Context	
> This function can be called from task level only.	

Table 5-27 Cry_ HmacSha1VerifyMainFunction

5.3.22 Cry_RsaDecrypt1024Start

Prototype	
`void` **`Cry_RsaDerypt1024Start`** `(Const void *cfgPtr, const Csm_AsymPrivateKeyType *keyPtr)`	
Parameter	
cfgPtr	Pointer to ConfigStructure
keyPtr	Holds a pointer to the key which has to be used during the asymmetrical decryption operation.
Return code	
CSM_E_OK	Request successful.
CSM_E_NOT_OK	Request failed.
Functional Description	
This interface shall be used to initialize the asymmetrical decryption.	
Particularities and Limitations	
> This function can be synchronous or asynchronous. > This function is non-reentrant. > This function is called by application. **Caution** The application (SWC) should pass a pointer to a structure of type Cry_Rsa1024KeyType containing the RSA private key. The pointer to this structure has to be casted to Csm_AsymPrivateKeyType in order to match the API.	
Call Context	
> This function can be called from task level only.	

Table 5-28 Cry_ RsaDecrypt1024Start

5.3.23 Cry_RsaDecrypt1024Update

Prototype	
`void` **`Cry_RsaDerypt1024Update`** `(void)`	
Parameter	
cfgPtr	Pointer to ConfigStructure
cipherTextPtr	Holds a pointer to the encrypted data.
cipherTextLenght	Contains the length of the encrypted data in bytes
plainTextPtr	Holds a pointer to the memory location which will hold the decrypted text.
plainTextLenght	Holds a pointer to a memory location in which the length information is stored. On calling this function this parameter shall contain the size of the buffer provided by plainTextPtr. When the request has finished, the amount of data that has been decrypted shall be stored.
Return code	
CSM_E_OK	Request successful.
CSM_E_NOT_OK	Request failed.
CSM_E_SMALL_BUFFER	The provided buffer is too small to store the result.
Functional Description	
This interface shall be used to feed the asymmetrical decryption with input data.	
Particularities and Limitations	
> This function can be synchronous or asynchronous. > This function is non-reentrant. > This function is called by application.	
Call Context	
> This function can be called from task level only.	

Table 5-29 Cry_ RsaDecrypt1024Update

5.3.24 Cry_RsaDecrypt1024Finish

Prototype	
`void` **`Cry_RsaDerypt1024Finish`** `(void)`	
Parameter	
cfgPtr	Pointer to ConfigStructure
plainTextPtr	Holds a pointer to the memory location which will hold the decrypted text.
plainTextLenght	Holds a pointer to a memory location in which the length information is stored. On calling this function this parameter shall contain the size of the buffer provided by plainTextPtr. When the request has finished, the amount of data that has been decrypted shall be stored.
Return code	
CSM_E_OK	Request successful.
CSM_E_NOT_OK	Request failed.
CSM_E_SMALL_BUFFER	The provided buffer is too small to store the result.
Functional Description	
This interface shall be used to finish the asymmetrical decryption.	
Particularities and Limitations	
> This function can be synchronous or asynchronous. > This function is non-reentrant. > This function is called by application.	
Call Context	
> This function can be called from task level only.	

Table 5-30 Cry_ RsaDecrypt1024Finish

5.3.25 Cry_RsaDecrypt1024MainFunction

Prototype	
`void` **`Cry_ RsaDecrypt1024MainFunction`** `(void)`	
Parameter	
-	
Return code	
-	
Functional Description	
This function implements the asynchronous service handling. **Note** This function is empty if ‘Use Sync Job Processing’ is enabled.	
Particularities and Limitations	
> This function is synchronous. > This function is not reentrant. > This function has to be called by CSM. > This function must not be called by the application.	
Call Context	
> This function can be called from task level only.	

Table 5-31 Cry_ RsaDecrypt1024MainFunction

5.3.26 Cry_RsaSha1SigVerStart

Prototype	
`void` **`Cry_RsaSha1SigVerStart`** `(Const void *cfgPtr, const Csm_AsymPublicKeyType *keyPtr)`	
Parameter	
cfgPtr	Pointer to ConfigStructure
keyPtr	Holds a pointer to the key necessary for the signature verification operation.
Return code	
CSM_E_OK	Request successful.
CSM_E_NOT_OK	Request failed.
Functional Description	
This interface shall be used to initialize the signature verification.	
Particularities and Limitations	
> This function can be synchronous or asynchronous. > This function is non-reentrant. > This function is called by application. **Caution** The application (SWC) should pass a pointer to a structure of type Cry_RSASigKeyType containing the RSA public key. The pointer to this structure has to be casted to Csm_AsymPublicKeyType in order to match the API.	
Call Context	
> This function can be called from task level only.	

Table 5-32 Cry_ RsaSha1SigVerStart

5.3.27 Cry_RsaSha1SigVerUpdate

Prototype	
`void` **`Cry_RsaSha1SigVerUpdate`** `(void)`	
Parameter	
cfgPtr	Pointer to ConfigStructure
dataPtr	Holds a pointer to the signature which shall be verified.
dataLength	Contains the length of the signature to verify in bytes
Return code	
CSM_E_OK	Request successful.
CSM_E_NOT_OK	Request failed.
Functional Description	
This interface shall be used to feed the signature verification with input data.	
Particularities and Limitations	
> This function can be synchronous or asynchronous. > This function is non-reentrant. > This function is called by application.	
Call Context	
> This function can be called from task level only.	

Table 5-33 Cry_ RsaSha1SigVerUpdate

5.3.28 Cry_RsaSha1SigVerFinish

Prototype	
`void` **`Cry_RsaSha1SigVerFinish`** `(void)`	
Parameter	
cfgPtr	Pointer to ConfigStructure
signaturePtr	Holds a pointer to the memory location which holds the signature to be verified.
signatureLength	Holds the length of the signature to be verified.
resultPtr	Holds a pointer to the memory location which will hold the result of the signature verification.
Return code	
CSM_E_OK	Request successful.
CSM_E_NOT_OK	Request failed.
Functional Description	
This interface shall be used to finish the signature verification.	
Particularities and Limitations	
> This function can be synchronous or asynchronous. > This function is non-reentrant. > This function is called by application.	
Call Context	
> This function can be called from task level only.	

Table 5-34 Cry_ RsaSha1SigVerFinish

5.3.29 Cry_RsaSha1SigVerMainFunction

Prototype	
`void` **`Cry_ RsaSha1SigVerMainFunction`** `(void)`	
Parameter	
-	
Return code	
-	
Functional Description	
This function implements the asynchronous service handling. **Note** This function is empty if 'Use Sync Job Processing' is enabled.	
Particularities and Limitations	
> This function is synchronous. > This function is not reentrant. > This function has to be called by CSM. > This function must not be called by the application.	
Call Context	
> This function can be called from task level only.	

Table 5-35 Cry_ RsaSha1SigVerMainFunction

5.4 Services used by CRY

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

Component	API
CSM	Csm_<Service>CallbackNotification Csm_<Service>ServiceFinishNotification
SecMod[2]	Provided Service APIs

Table 5-36 Services used by the CRY

5.5 Service Ports

The current implementation of the CRY does not support Service Ports.

[2] Name of the module may differ

6 Configuration

In the CRY the attributes can be configured with the following tools:

> Configuration in DaVinci Configurator 5

6.1 Configuration Variants

The CRY supports the configuration variants

> `VARIANT-PRE-COMPILE`

6.2 Configuration with DaVinci Configurator 5

6.2.1 Common Properties

Attribute Name	Values Default value is typed bold	Description
CryUseSyncJobProcessing	**STD_ON** STD_OFF	Pre-processor switch to enable and disable synchronous job processing.
CryVersionInfoApi	**STD_ON** STD_OFF	Pre-processor switch to enable and disable availability of the API Cry_GetVersionInfo(). True: API Cry_GetVersionInfo() is available. False: API Cry_GetVersionInfo() is not available.

6.2.2 AES Encrypt/Decrypt Properties

Attribute Name	Values	Description
CryAes<Encrypt/Decrypt>128BlockMode	**CRY_AESBLOCKMODE_CBC** CRY_AESBLOCKMODE_ECB	The block mode describes how to handle data which exceeds the block length.
CryAes<Encrypt/Decrypt>128PaddingMode	**CRY_AESPADDINGMODE_PKCS5**	To align the data length to the block size a padding mode is required.

6.2.3 FIPS-186-2 Properties

Attribute Name	Values	Description
CrySaveState	**STD_ON** STD_OFF	For development, testing purposes and special use-cases the pseudo random number generator provides the option to deactivate the save state. Disabling this feature will produce the same result for each call of Csm_RandomGenerate until the seed is updated by the Csm_RandomSeed service.

7 AUTOSAR Standard Compliance

7.1 Deviations

The current implementation does not have any deviations.

7.2 Additions/ Extensions

The current implementation does not have any extensions.

7.3 Limitations

7.3.1 Support of Cryptographic Services

The current cryptographic services are supported:

▶ AES128 - Service for Symmetrical Interface
▶ FIPS-186 – Service for Random Interface
▶ HMAC SHA-1 – Service for MAC Interface
▶ RSA Decrypt - Service for Asymmetrical Interface
▶ RSA-SHA1 Signature Verification - Service for Signature Interface

Table 7-1 Supported AUTOSAR standard conform features

The following cryptographic services are not supported yet:

▶ Service for Hash Interface
▶ Service for Symmetrical Block Interface
▶ Service for Checksum Interface
▶ Service for Key Derivation Interface
▶ Service for Key Exchange Interface
▶ Service for Symmetrical Key Exchange Interface
▶ Service for Symmetrical Key Extract Interface
▶ Service for Asymmetrical Key Extract Interface

8 Glossary and Abbreviations

8.1 Glossary

Term	Description
Cryptographic Primitive	An underlying cryptographic module or library

Table 8-1 Glossary

8.2 Abbreviations

Abbreviation	Description
API	Application Programming Interface
AUTOSAR	Automotive Open System Architecture
BSW	Basis Software
CRY	Cryptographic library module
CSM	Crypto Service Manager
DEM	Diagnostic Event Manager
DET	Development Error Tracer
ECU	Electronic Control Unit
HIS	Hersteller Initiative Software
MICROSAR	Microcontroller Open System Architecture (the Vector AUTOSAR solution)
RTE	Runtime Environment
SchM	Schedule Manager
SRS	Software Requirement Specification
SWC	Software Component
SWS	Software Specification

Table 8-2 Abbreviations

9 Contact

Visit our website for more information on

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

www.vector.com