Elektrobit

EB tresos® AutoCore Generic 8 CRY documentation

Module release 2.6.6

Elektrobit Automotive GmbH
Am Wolfsmantel 46
91058 Erlangen, Germany
Phone: +49 9131 7701 0
Fax: +49 9131 7701 6333
Email: info.automotive@elektrobit.com

Technical support

https://www.elektrobit.com/support

Legal disclaimer

Confidential information.

Table of Contents

1. Overview

Welcome to the EB tresos AutoCore Generic 8 CRY product release notes and documentation.

This document provides:

- Chapter 2, “Cry module release notes“: details of changes and new features in the current release
- Chapter 3, “Cry user's guide“: concept information and configuration instructions
- Chapter 4, “Cry module references“: configuration parameters and the application programming interface

2. Cry module release notes

- AUTOSAR R4.0 Rev 3
- AUTOSAR SWS document version: 1.2.0
- Module version: 2.6.6.B402799
- Supplier: Elektrobit Automotive GmbH

2.1. Change log

This chapter lists the changes between different versions.

Module version 2.6.6

2020-02-18

- Implemented new feature which allows to configure the time spent in a main function call, during an AES Decryption and Encryption.

Module version 2.6.5

2019-11-19

- Improvement for Cry make files

Module version 2.6.4

2019-10-25

- ASCCRY-346 Fixed known issue: Cry module fails compilation if RSA SSA PSS signature verification is used with Multi compiler
- ASCCRY-355 Fixed known issue: SSG seed generation primitive overrides application data in RAM

Module version 2.6.3

2018-10-11

- Implemented speed optimized CRC algorithm

Module version 2.6.2

2018-07-26

- ASCCRY-335 Fixed known issue: EdDSA signature verification gets stuck in endless loop if the key/signature cannot be decoded

Module version 2.6.1

2018-06-22

- Corrected macro name for multiple inclusion protection for files Cry_HMACGen.h and Cry_HMACVrfy.h

Module version 2.6.0

2018-03-30

- Changed the HMAC primitive with respect to the generated/verified MAC length. The length of the generated MAC is now specified in bytes, whilst the length of the MAC to be verified is now specified in bits
- ASCCRY-323 Fixed known issue: RsaSsaV15 signature generation primitive gets stuck if synchronous mode is used
- ASCCRY-311 Fixed known issue: EdDSA signature generate/verify primitives get stuck if synchronous mode is used
- ASCCRY-307 Fixed known issue: Cry HMAC primitive calls the Csm callback notification with a wrong error code
- Improved the speed of the Long Number library
- ASCCRY-308 Fixed known issue: HMAC generate/verify primitives get stuck if synchronous mode is used
- ASCCRY-313 Fixed known issue: Elliptic curve key extraction primitive gets stuck if synchronous mode is used
- ASCCRY-312 Fixed known issue: CCM encryption/decryption primitives get stuck if synchronous mode is used

Module version 2.5.0

2018-01-30

- ASCCRY-298 Fixed known issue: Cry will not compile if only CCM encryption is used
- ASCCRY-294 Fixed known issue: RSA signature verify primitives get stuck if synchronous mode is used
- Implemented the time slicing feature for EdDSA signature generation/verification

- ASCCRY-301 Fixed known issue: Cry symmetric block encryption/decryption primitive gets stuck if the callback is called with an error code

Module version 2.4.0

2017-11-16

- ASCCRY-279 Fixed known issue: Cry might not compile if a mix between RsaSsaV15 and RsaSsaPss signature verifications with Barrett reduction is used
- Implemented performance optimized RSA verification variants
- Improved the EdDSA signature verification algorithm
- Implemented elliptic curve asymmetric public key extraction algorithm
- Optimized the speed of the SHA-1/SHA-2 algorithm implementations. Implemented non-interruptable SHA-1/SHA-2 algorithm variants
- ASCCRY-266 Fixed known issue: RsaSsaV15 signature generation primitive might perform an out-of-bounds memory access

Module version 2.3.0

2017-09-28

- Implemented checksum algorithms:CRC-8, CRC-16, CRC-32.
- Optimized the speed of the AES algorithm implementation. Implemented a look-up table based AES encryption and decryption variant
- Implemented the immediate restart feature for the following primitives: It is available for the MD5, SHA-2, SHA-3, CMAC, HMAC, SSGSeed, SSGGenerate, AES encryption/decryption, CBC PKCS7 encryption/decryption, RSA and symmetric key extraction, RSASSA-PSS and RSASSA-PKCS1-v1_5 signature verification.
- Implemented RSASSA-PKCS1-v1_5 signature generation algorithm.
- Implemented SHA-224, SHA-384 and SHA-512 algorithms.
- Implemented feature which allows to configure the number of Cry_SHA1 or Cry_SHA2 algorithms iterations per main function call.
- Implemented OMAC2 and CCM algorithm.
- Implemented EdDSA signature generation/verification algorithm.
- Implemented the immediate restart feature for the EdDSA signature generation/verification.
- Implemented feature to restart an ongoing cryptographic process. The feature was added for the HashEdDSA variants of Edwards-curve Digital Signature Algorithm, which is named Ed25519ph.

Module version 2.2.0

2017-07-03

- Implemented asymetric public key extraction algorithm.
- Added salt length check for RsaSsa-PSS siganture verification.
- Implemented MD5 hash functions.
- ASCCRY-228 Fixed known issue: Non-existing functions referenced in Cry_Bswmd.arxml
- ASCCRY-229 Fixed known issue: BSW-MODULE-ENTRY for SHA3 is missing
- ASCCRY-230 Fixed known issue: Invalid BSW-MODULE-ENTRY Cry_CMACGentart referenced in Cry_-Bswmd.arxml

Module version 2.1.0

2017-03-31

- Implemented SHA-3 hash functions.
- Implemented Edwards-curve Digital Signature Algorithm.
- Implemented Edwards-curve Digital Signature Algorithm.

Module version 2.0.3

2016-06-22

- ASCCRY-132 Fixed known issue: Incorrect closing of memory section in file Cry_CVCPublicKeyExtract.c
- Implemented new feature which allows to configure the time spent in a main function call, during signature verification (RsaSsaPss and RsaSsaV15). Added feature that allows configuring time slicing callback function, when time slicing is activated.
- Implemented feature to cancel an ongoing cryptographic process. The feature was added for SHA-2, RSASSA-PKCS1-v1_5 and RSASSA-PSS primitives.

Module version 2.0.2

2016-04-27

- Improved stability of Der parser for primitive `CVCPublicKeyExtract`
- Corrected parameter description of the application programming interface (API)
- Added new configuration parameter `CryCVCPublicKeyExtractEnableCertChain`

- Changed the return value of the API `Cry_CMACGenFinish` from CSM_E_SMALL_BUFFER to CSM_E_NOT_OK
- ASCCRY-135 Fixed known issue: Cry_RsaSsaV15VerifyUpdate and Cry_RsaSsaV15VerifyFinish return CSM_E_BUSY when CSM_E_NOT_OK is expected
- ASCCRY-136 Fixed known issue: If Csm_HashUpdate returns an error, the algorithm Cry_RsaSsaV15 does not report the error to the application

Module version 2.0.0

2015-10-15

- Cry primitives compliant to AUTOSAR 4.0.3

2.2. New features

2.3. EB-specific enhancements

This chapter lists the enhancements provided by the module.

- This module provides no EB-specific enhancements.

2.4. Deviations

This chapter lists the deviations of the module from the AUTOSAR standard.

- Synchronous job processing not supported

  Description:

  Synchronous job processing as specified by AUTOSAR is not supported by the Cry module.

  Requirements:

  CSM0031, CSM0035, CSM0505
- API functions do not return CSM_E_SMALL_BUFFER

  Description:

The CSM specification only defines a reaction for the internal CSM state for the API functions for the following CRY primitive return values: CSM_E_OK, CSM_E_BUSY, CSM_E_NOT_OK. Therefore the CRY shall check if the provided buffer is large enough to hold the result of the computation. If the provided buffer is too small, CSM_E_NOT_OK shall be returned. See http://www.autosar.org/bugzilla/show_bug.cgi?id=71250 for more information. Affected APIs: `Cry_CMACGenFinish`, `Cry_AES_DecryptUpdate`, `Cry_AES_EncryptUpdate`.

Requirements:

CSM0663, CSM0700, CSM0662, CS_SECURE_CRY_0042, CS_SECURE_CRY_0038

2.5. Limitations

This chapter lists the limitations of the module. Refer to the module references chapter *Integration notes*, subsection *Integration requirements* for requirements on integrating this module.

- The CCM algorithm does not support the streaming approach

  Description:

  The CCM algorithm provides support for Authenticated Encryption with Associated Data. As a consequence to the fact that this algorithm first authentificates the data and then encrypts it all input data must be provided with a single call of `Csm_<Service>Update`.

  In the case of decryption the plain text should be available only after the MAC is verified.

- The CCM algorithm supports only payloads smaller than 2^32 bytes

  Description:

  The maximum size of the payload data must be lower than 2^32 bytes.

- The EdDSA algorithm does not support configurable context

  Description:

  Irtf-cfrg-eddsa-08 RFC specifies in Chapter 5.1. that Ed25519ph provides support for configurable context.

  In the current implementation the context is always empty.

2.6. Open-source software

Open-source software information is not available for this module.

3. Cry user's guide

3.1. Overview

This user's guide describes the Cryptographic Primitives (Cry) module and explains the basic functionality of the Cry. The guide also describes the modules which are necessary to configure the Cry module. The Cry module reference provides further information on configuring the Cry itself.

This user's guide is intended for readers who have good knowledge of AUTOSAR and about the purpose of the Cry. The information provided here should help you to integrate the Cry in your AUTOSAR project.

- Section 3.2, “Background information” provides an overview of the basic functionality of the Cry.
- Section 3.3, “Configuring the Cry” provides information on related modules that are needed in order to configure the Cry.
- For configuration details for the Cry module itself, see the parameter descriptions provided in the Cry module reference Chapter 4, “Cry module references“.

3.2. Background information

The Cry module provides several cryptographic functionalities. Cry acts as a lower interface to the Csm software module, which in turn provides access to the cryptographic functionalities to the user via services.

The Cry module is implemented according to the following specifications:

- AUTOSAR 4.0.3 Specification of Crypto Service Manager, v1.2.0
- Software Specification Cry, Conti VED Cyber Security, v1.0.0

The primitives of the Cry module are provided as asynchronous functions.

You need a Csm module to use the cryptographic functionality provided by the Cry module. You must create configurations in the Csm which link to the Cry configurations.

3.2.1. Mapping of Cryptographic Functionalities to Csm Services

The following is a list of the cryptographic primitives which are priovided by the Cry. It also contains the string which has to be entered into the field "Primitive name" in the Csm configuration created for the service.

Functionality	Csm Service	Cry Primitive Name
symmetrical key extraction	SymKeyExtract	KEY_SYM_Extract
AES-ECB encryption	SymBlockEncrypt	AES_Encrypt
AES-ECB decryption	SymBlockDecrypt	AES_Decrypt
CBC decryption with PKCS7 Padding	SymDecrypt	CbcPkcs7Decrypt
CBC encryption with PKCS7 Padding	SymEncrypt	CbcPkcs7Encrypt
CCM encryption	SymEncrypt	CCMEncrypt
CCM decryption	SymDecrypt	CCMDecrypt
CMAC/OMAC2 generation	MacGenerate	CMACGen
CMAC/OMAC2 verification	MacVerify	CMACVrfy
CV certificate key extraction	AsymPublicKeyExtract	CVCPublicKeyExtract
RSASSA-PSS signature verification	SignatureVerify	RsaSsaPssVerify
RSASSA-PKCS1-v1_5 signature verification	SignatureVerify	RsaSsaV15Verify
EdDSA signature generation	SignatureVerify	EdDSAGen
EdDSA signature verification	SignatureGenerate	EdDSAVrfy
SHA-1 Hash	Hash	SHA1
SHA2-224 Hash	Hash	SHA2
SHA2-256 Hash	Hash	SHA2
SHA2-384 Hash	Hash	SHA2
SHA2-512 Hash	Hash	SHA2
SHA3-224 Hash	Hash	SHA3
SHA3-256 Hash	Hash	SHA3
SHA3-384 Hash	Hash	SHA3
SHA3-512 Hash	Hash	SHA3
MD5 Hash	Hash	MD5
Pseudo random generation using the Self-Shrinking Generator	RandomGenerate	SSGGenerate
Seeding the pseudo random number generator (Self-Shrinking Generator)	RandomSeed	SSGSeed

Functionality	Csm Service	Cry Primitive Name
RSASSA-PKCS1-v1_5 signature generation	Csm_SignatureGenerate	RsaSsaV15Generate
RSA key extraction	AsymPublicKeyExtract	ExtractRsaPublicKey
Elliptic curve key extraction	AsymPublicKeyExtract	ExtractECPubKey

Table 3.1. Cry primitives

3.2.2. Using the time slicing feature for RSASSA-PSS and RSASSA-PKCS1-v1_5 signature verification

This feature is used to limit the maximum duration of a main function call and to conform to the real-time requirements of the system. It is the trade-off between the duration of one main function call on one other hand and the overall duration of a signature verification, on the other hand. You can activate the feature by enabling the parameters `CryRsaSsaPssVerifyUseTimeSlices` and `CryRsaSsaV15VerifyUseTimeSlices`. Furthermore, the number of the time slices should be configured by setting the parameters `CryRsaSsaPssVerifyNumberOfTimeSlices` and `CryRsaSsaV15VerifyNumberOfTimeSlices`.

You can configure an optional callback function name for this feature, which returns the maximum number of iterations for the use during long-running calculations during signature a verification. The callback function can used to facilitate the evaluation of the correct number of time slices. This feature is activated by enabling `CryRsaSsaPssVerifyUseCbk` and `CryRsaSsaV15VerifyUseCbk`. The names of the callback functions are configured using the parameters `CryRsaSsaPssTimeSlicesCbk` and `CryRsaSsaV15TimeSlicesCbk`.

3.2.3. Using the restart feature

The restart feature allows to restart an already running calculation for signature verification and hash calculation. The precondition to use this service is the usage of a CSM which also supports the restart feature. When a `Csm_<Service>Start` function is called when the service is not idle, and the `Csm_<Service>Start` is called with the same `cfgId` than the currently running calculation, the CSM shall restart the service by calling the `Cry_<Service>Start`.

The restart feature can be enabled with configuration parameter `Cry<Primitive>SupportRestart`. It is available for the SHA-256 and all SHA-3 hashes, RSA, RSASSA-PSS and RSASSA-PKCS1-v1_5 signature verification, as well as EdDSA signature generation and verification.

NOTE **Restart Feature and Callbacks**

When you use the restart feature, please note that a service should only be restarted after the last expected callback notification was called. Otherwise, if `Cry_<Service>Start` can interrupt `Cry_<Service>MainFunction`, callback notifications related to the canceled calculation might be called.

3.2.4. Using the immediate restart feature

The restart feature allows to restart an already running calculation regardless of the configuration ID or if other instances are running. The precondition to use this feature is the usage of a CSM which also supports the immediate restart feature. When a `Csm_<Service>Start` function is called the CSM shall restart the service by calling the `Cry_<Service>Start`.

The immediate restart feature can be enabled with configuration parameter `Cry<Primitive>ImmediateRestart`. It is available for the following algorithms:

- MD5
- SHA-2
- SHA-3
- CMAC
- HMAC
- SSGSeed
- SSGGenerate
- AES encryption/decryption
- CBC PKCS7 encryption/decryption
- RSA and symmetric key extraction
- RSASSA-PSS and RSASSA-PKCS1-v1_5 signature verification
- EdDSA signature generation and verification

NOTE **Immediate restart feature on complex primitives**

When you use the immediate restart feature on a primitive which needs a another primitive for part of processing both of them need to have the immediate restart feature enabled. For example, the RsaSsaV15 signature verification depends on the SHA primitive for it's internal calculations. In this case the SHA primitive and the corresponding CSM service need to have the Immediate Restart feature enabled.

3.2.5. AES implementation variants

The `Cry` module offers two implementation variants for the encryption and decryption parts of the AES algorithm. These variants allow you to achieve a balance between resource usage and processing speed according to your project's needs. The implementation variants are described below:

- `RESOURCE_EFFICIENT` - this implementation variant is designed to offer a balance between the processing time and resource usage.
- `FAST` - this implementation variant is based on pre-computed look-up tables and is designed to offer maximum processing speed by trading resource consumption.

Table 3.2, “Cry AES encryption-only RAM resource usage” provides approximate values for the RAM usage of the look-up tables for the implementation variants when only encryption is enabled.

Implementation variant (encryption)	RAM usage (bytes)
RESOURCE_EFFICIENT	1279
FAST	1576

Table 3.2. Cry AES encryption-only RAM resource usage

Table 3.3, “Cry AES decryption-only RAM resource usage” provides approximate values for the RAM usage of the look-up tables for the implementation variants when only decryption is enabled.

Implementation variant (decryption)	RAM usage (bytes)
RESOURCE_EFFICIENT	1279
FAST	1576

Table 3.3. Cry AES decryption-only RAM resource usage

Table 3.4, “Cry AES encryption and decryption RAM resource usage” provides approximate values for the RAM usage of the look-up tables for the implementation variants when both encryption and decryption are enabled.

Implementation variant (encryption and decryption)		RAM usage (bytes)
RESOURCE_EFFICIENT	RESOURCE_EFFICIENT	1279

Implementation variant (encryption and decryption)		RAM usage (bytes)
RESOURCE_EFFICIENT	FAST	2343
FAST	RESOURCE_EFFICIENT	2343
FAST	FAST	2600

Table 3.4. Cry AES encryption and decryption RAM resource usage

3.2.6. Using PZ1A Optimization for Cry AES

This feature is used to limit the maximum duration of a main function call and to conform to the real-time requirements of the system. It is the trade-off between the duration of one main function call on the one hand, and the overall duration of a Cry AES encryption/decryption, on the other hand.

You can activate the feature by enabling the parameters PZ1ACryOptimEncry or PZ1ACryOptimDecry.

If enabled, the execution of the main function is divided in small execution steps, allowing to have OS interruptions between them. As a consequence the total time execution of a Cry job will take longer, as it can be interupted by other running processes/tasks. This is useful when processing large amount of data with Cry (which takes a long time), enabling non-blocking operation of other processes/tasks.

This optimization impacts the following Cry Primitives:

- CCMEncrypt
- CCMDecrypt
- CMACGen (indirect impact)
- CMACVrfy (indirect impact)

3.2.7. Using the time slicing feature for SHA-1 and SHA-2 primitives

This feature is used for configuring the maximum number of SHA-1/SHA-2 algorithm iterations during the compression phase per `Cry_SHA1MainFunction()`/`Cry_SHA2MainFunction()` call in order to conform to the real-time requirements of the system. It is the trade-off between duration of one main function call and overall duration of a digest generation. The feature is activated by configuring the number of algorithm iterations per main function using the parameters `CrySHA1IterationsPerMain` or `CrySHA2IterationsPerMain` respectively. If zero is configured the number of internal iterations shall not be limited.

This feature is available only if SHA-1/SHA-2 interruptible code variant is used (`CrySHAOneAndTwoImplementation` = `CRY_SHAONEANDTWO_INTERRUPTABLE`).

3.2.8. Using the time slicing feature for EdDSA signature generation and verification primitives

This feature is used for configuring the maximum number of EdDSA signature generation/verification algorithm iterations during long number operations or point operations per `Cry_EdDSAGenMainFunction()`/`Cry_EdDSAVrfyMainFunction()` call in order to conform to the real-time requirements of the system. It is the trade-off between duration of one main function call and overall duration of a signature generation/verification. The feature is activated by configuring the number of algorithm iterations per main function using the parameters `CryEdDSAGenNumberOfTimeSlices` or `CryEdDSAVrfyNumberOfTimeSlices` respectively. If zero is configured, the number of internal iterations shall not be limited. If one is configured, the slicing mechanism is disabled.

3.2.9. Optimize for Cry CRC

The `CRC` primitive can be tailored to the project-specific resource consumption and performance needs by setting the values of the following configuration parameter:

- `CRY_CRC_MEMORY_OPTIMIZED`

  This implementation variant uses the least amount of RAM during the generating of CRC Lookup Table. The CRC Lookup Table is re-calculated for every initialization The stack usage during of this implementation variant is negligible.

  This variant is intented for the use on platforms that have a limited amount of RAM memory available for cryptographic algorithms.

- `CRY_CRC_SPEED_OPTIMIZED`

  This initialization variant is used to store the CRC Lookup table values in a constant variable in ROM beacause they don't change. This is designed to achieve the maximum processing speed.

  As an example this implementation requires 256 bytes * (number of CRC). For exemple 256 * 2 = 512 bytes for storing 2 CRC confiuguration.

  This variant is intented for the use on platforms that have a larger amount of available RAM for cryptographic algorithms.

3.2.10. Long number library implementation variants

The `Cry` module implements a customizable, long number library, which is used for all public-key algorithms, for example RSA signature generation/verification, EdDSA signature generation/verification. The long number

library can be tailored to the project-specific resource consumption and performance needs by setting the values of the following three configuration parameters:

- `CryLNAlgorithmImplementation`
  - `CRY_LN_MEMORY_OPTIMIZED`

    This implementation variant uses the least amount of RAM during the processing of long number operations. The stack usage during of this implementation variant is negligible.

    This variant is intented for the use on platforms that have a limited amount of RAM memory available for cryptographic algorithms.
  - `CRY_LN_SPEED_OPTIMIZED`

    This implementation variant is designed to achieve the maximum processing speed at the cost of requiring an increased stack usage for storing the intermediate calculation results. An estimate value in bytes of the least amount of required stack-allocated RAM can be calculated by multiplying the byte length of the largest private/public key by `4`.

    As an example, a 1024 bit key, requires a stack usage of at least (1024 bits / 8) * 4 = 512 bytes for storing the intermediate results only.

    This variant is intented for the use on platforms that have a larger amount of available RAM for cryptographic algorithms.
- `CryInterruptableLN`

  This configuration parameter is available only if `CRY_LN_MEMORY_OPTIMIZED` algorithm implementation variant is used.
  - `true`

    The `Cry` module splits the long number operations into several sub-functions, which are executed asynchronously, during `Cry` main function calls.

    This implementation variant is intented for the use in projects which require a low CPU usage during long number operations.
  - `false`

    The `Cry` module performs long number operations synchronously, in a single function call.

    This implementation variant is intented for the use in projects which require a lower total processing time of the algorithm and can afford a high CPU usage during long number operations.
- `CryLNPlatformDoubleWordSupport`

  This configuration parameter is available only if `CRY_LN_SPEED_OPTIMIZED` algorithm implementation variant is used.

The Cry module stores long numbers in arrays of Cry_LNWordType elements, e.g. used to represent the value of the modulo from public keys. These elements have the length equal to the target platform's word length. During calculations involving long numbers, the partial results can exceed the precision of the Cry_LNWordType. The value of the CryLNPlatformDoubleWordSupport configuration parameter defines the handling of operations performed on Cry_LNWordType elements.

- true

  The Cry module defines and uses variables of type Cry_LNDoubleWordType, which are twice as large as Cry_LNWordType. For example, if the target platform has a 32 bit CPU, Cry_LNWordType is represented using a 32 bit unsigned integer while Cry_DoubleLNWordType is represented using a 64 bit unsigned integer.

  The target platform and compiler need to offer support for data types that have twice the length of the platform's word length.

  This implementation variant is intented for the use in projects which require the shortest processing time for cryptographic algorithms.

- false

  The Cry module splits operations on Cry_LNWordType elements into operations on the corresponding lower/upper parts of the word.

  This implementation variant is intented for the use on targets which do not offer support for data types larger than the platform's word length.

Figure 3.1, “Cry long number library implementation variants” summarizes the implementation variants of the long number library and shows the increase of processing speed and RAM usage according to the value of the three configuration parameters previously described.

Figure 3.1. Cry long number library implementation variants

3.3. Configuring the Cry

The following section explains how to configure a signature verification service using the RSASSA-PSS algorithm with a SHA-256.

- Open the Cry module configuration.
- Switch to the tab of the primitive SHA2 to configure.
- Click the **Add element** button to create a primitive configuration.

Figure 3.2. Cry SHA tab

- Open the `Csm` module configuration.
- Switch to the tab of the service corresponding to the `Cry` primitive.
- Click the **Add element** button to create a service configuration.
- The value of the `InitConfiguration` must be the name of the configuration created in the `Cry`.
- The value of the `PrimitiveName` must be the name of the corresponding `Cry` primitive.
- The value of the `Callback` is the name of the function, which is invoked by the Csm, when the service has finished a calculation step.

Figure 3.3. Csm Hash tab

- Open the `Cry` module configuration again.
- Switch to the tab of primitive RsaSsaPssVerify to configure.
- Now, the Hash service configuration created in the `Csm` can be referenced.
- For some primitives a key length can be configured.
- Click the **Add element** button to create a primitive configuration.

Figure 3.4. Cry RsaSsaPssVerify tab

- Open the `Csm` module configuration again.
- Switch to the tab of the service corresponding to the `Cry` primitive.
- Click the **Add element** button to create a service configuration.
- The value of the `InitConfiguration` must be the name of the configuration created in the `Cry`.
- The value of the `PrimitiveName` must be the name of the corresponding `Cry` primitive.
- The value of the `Callback` is the name of the function, which is invoked by the Csm, when the service has finished a calculation step.

Figure 3.5. Csm Signature Verify tab

- Switch to the `Csm` General tab.
- In the respective service container a key length can be configured.

  Configure the length of the key large enough to hold the key configured for the corresponding `Cry` primitive.

Figure 3.6. Csm General tab

To verify the signature of some data, the user application needs to implement the following call sequence:

- First call `Csm_SignatureVerifyStart()` and pass the `Csm` Signature Verify configuration and a key to the function.
- While the `Csm_MainFunction()` is schedulded cyclically wait for the application's callback to be invoked.
- Then call `Csm_SignatureVerifyUpdate()` and pass the `Csm` Signature Verify configuration and the signed data to the function.
- While the `Csm_MainFunction()` is schedulded cyclically wait for the application's callback to be invoked.
- The previous two steps might be repeated if the data is passed to the services in several chunks.
- Then call `Csm_SignatureVerifyFinish()` and pass the `Csm` Signature Verify configuration, the signature and a buffer for the verification result to the function.
- While the `Csm_MainFunction()` is schedulded cyclically wait for the application's callback to be invoked.
- Now, the provided buffer holds the verification result.

For simplicity, the given sequence does not take other return values than CSM_E_OK into account. The application must be able to react to other values returned by the APIs or passed to the Callback function

3.4. Signature verify key for RSA

The `Csm` module requires a key to perform signature verifications.

The following code snippet describes the format of the public key used by the EB crypto library and how to initialize the content of the data structure containing the public key. The length of the "Public exponent", "Modulus" and "Barrett factor" words is defined by the value of the `CryExtractRsaPublicKeyLength` configuration parameter.

```
const Csm_AsymPublicKeyType SignatureVerifyKeyPtr =
{
  0U,       /* Dummy length element for compatibility with Csm_AsymPublicKeyType */
  {
    /* Public exponent */
    PublicExponentNumWords,  /* Length of the public key exponent in words */
    PublicExponentWord1,     /* Least significant bytes of the public exponent */
    ...,
    PublicExponendWordP,     /* Most significant bytes of the public exponent */
```

```
    /* Modulus */
    ModulusNumWords,          /* Length of the public key modulus in words */
    ModulusWord1,             /* Least significant bytes of the modulus */
    ModulusWord2,
    ...,
    ModulusWordM,             /* Most significant bytes of the modulus */

    /* Barrett factor */
    BarrettNumWords,          /* Length of the public key Barrett factor in words */
    BarrettWord1,             /* Least significant bytes of the Barrett factor */
    BarrettWord2,
    ...,
    BarrettWordN              /* Most significant bytes of the Barrett factor */
  }
}
```

NOTE

Public exponent value

The value of the Public exponent is defined as 17 (0x11).

Make sure that when you initialize the `SignatureVerifyKeyPtr` structure, after setting the length and value of the Public exponent, you fill in the remaining bytes up until the "Modulus" with 0 (`CRY_RSAPUBKEY_NUM_LEN_WORDS` minus 2).

To fill-in the content of the public key data structure, you have to follow the sequence of steps described below:

1. Extract the content of the certificate (`.cer` file) which stores the public key (comprised of the "Public exponent", "Modulus" and "Barrett factor") using an ASN.1 decoder/parser
2. Remove the leading zero(s) from the "Public exponent", "Modulus" and "Barrett factor" of the values extracted at step 1
3. Proceed with one of the following two approaches:
   - Use the asymmetrical key extract interfaces provided by the `Csm`. To configure the asymmetrical key extract service, take the following steps:
     a. Open the editor of the `Cry` module and open the tab **ExtractRsaPublicKey**.
     b. Add a new entry to the row and configure the parameters as shown in Figure 3.7, “ExtractRsaPublicKey configuration for `SecDiag` in `Cry`”.

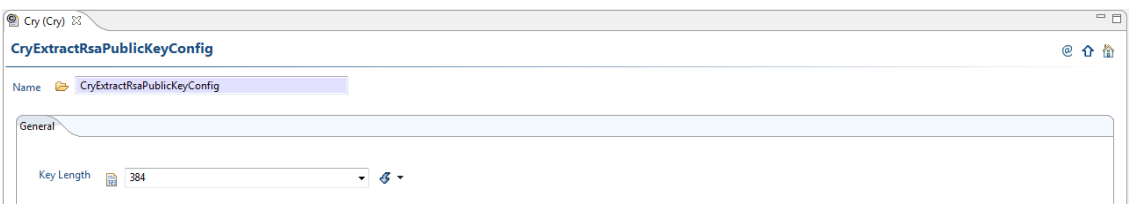

Figure 3.7. ExtractRsaPublicKey configuration for `SecDiag` in `Cry`

The name of the Cry ExtractRsaPublicKey configuration can be freely chosen. The value of the Key Length parameter needs to be set to 385.

c. Open the editor of the Csm module. In the **CsmAsymPublicKeyExtract** container, set the value of the Maximal key size parameter to 1200.

d. Open the **AsymPublicKeyExtract** tab of the Csm module.

e. Add a new entry to the row and configure the parameters as show in Figure 3.8, “AsymPublicKeyExtract configuration for SecDiag in Csm”.

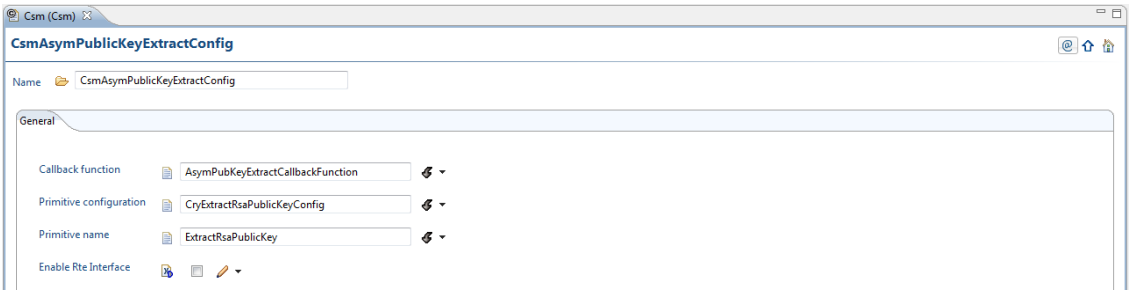

Figure 3.8. AsymPublicKeyExtract configuration for SecDiag in Csm

The name of the Csm AsymPublicKeyExtract configuration can be freely chosen. The value of the Callback function can be freely chosen and must be set to a user-defined function. This function will be called when the Csm module is finished extracting the public key. The value of the Primitive configuration parameter must match the name of the Cry ExtractRsaPublicKey configuration added in the first step. The value of the Primitive name parameter must be set to ExtractRsaPublicKey.

You have to call the Csm_AsymPublicKeyExtractStart(), Csm_AsymPublicKeyExtractUpdate() and Csm_AsymPublicKeyExtractFinish() APIs from integration code using the ID of the newly added Csm AsymPublicKeyExtract configuration.

The Csm_AsymPublicKeyExtractUpdate() API expects the input key as an uint8 array of the following form:

```
uint8 inputKey[] =
{
  /* 384 Byte Modulus */,
  /* 384 Byte Barrett */,
  /* 384 Byte Public Exponent */
};
```

Copy the values of the "Public exponent", "Modulus" and "Barrett factor" from step 2 into `inputKey[]` array, taking into consideration the order of the public key elements. The Public Exponent needs to be padded with leading zeros.

The `Csm_AsymPublicKeyExtractFinish()` API expects an output pointer of type `Csm_AsymPublicKeyType` (compatible to `Cry`'s internal `CryAsymPublicKeyType` type) to store the extracted key.

- Split the values of the "Public exponent", "Modulus" and "Barrett factor" obtained at step 2 into platform architecture sized words and set the content of the arrays in a little-endian word-order.

Public key example

The marked values from Figure 3.9, “Public key modulus” depict the key's "Modulus", as extracted from the `.cer` file. Note the value `0` padding (leading byte), which has to be stripped.

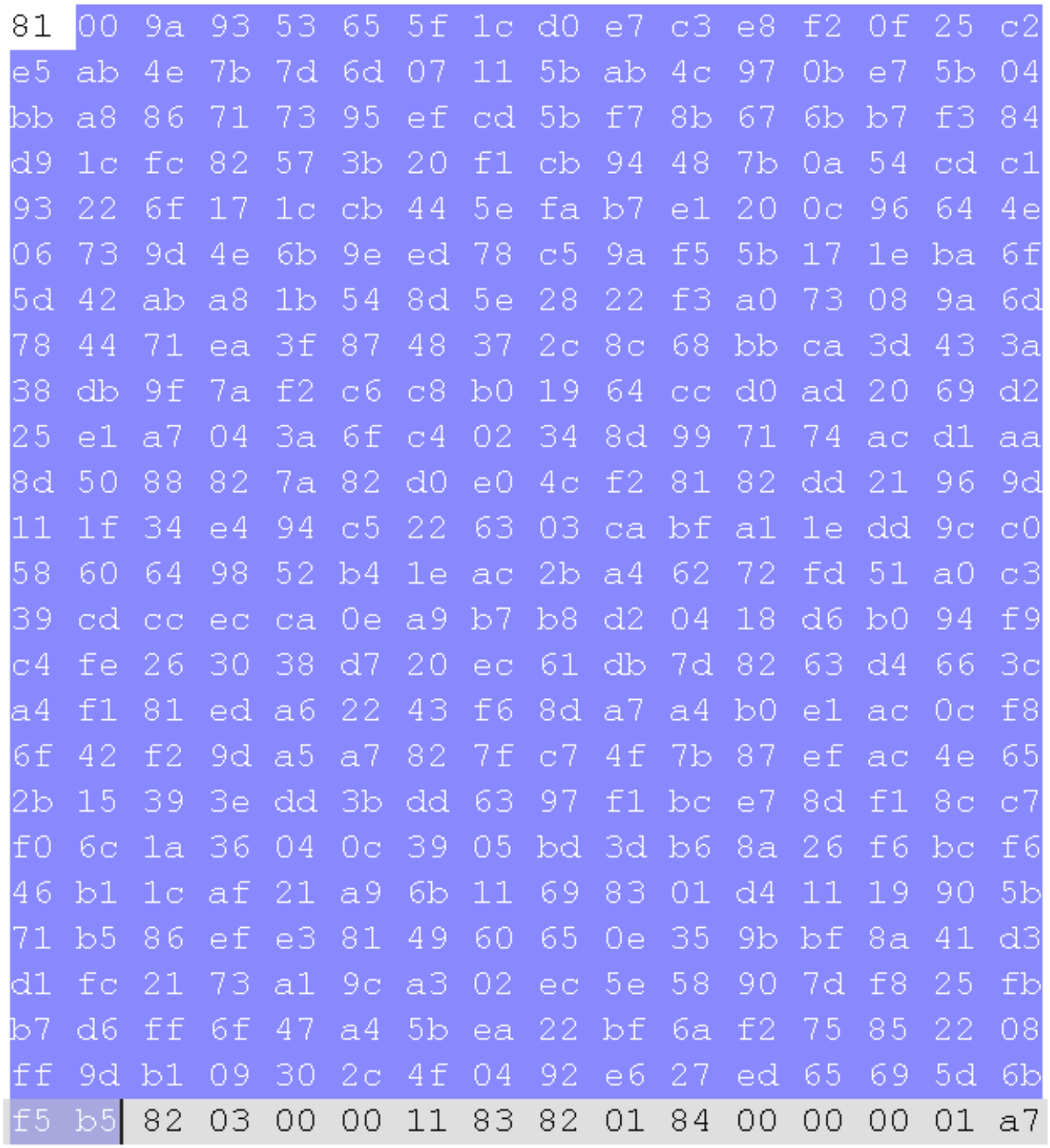

Figure 3.9. Public key modulus

The marked values from Figure 3.10, “Public key modulus” depict the key's "Public exponent", as extracted from the `.cer` file. Note the value `0` padding (leading bytes), which have to be stripped.

```
f0 6c 1a 36 04 0c 39 05 bd 3d b6 8a 26 f6 bc f6
46 b1 1c af 21 a9 6b 11 69 83 01 d4 11 19 90 5b
71 b5 86 ef e3 81 49 60 65 0e 35 9b bf 8a 41 d3
d1 fc 21 73 a1 9c a3 02 ec 5e 58 90 7d f8 25 fb
b7 d6 ff 6f 47 a4 5b ea 22 bf 6a f2 75 85 22 08
ff 9d b1 09 30 2c 4f 04 92 e6 27 ed 65 69 5d 6b
f5 b5 82 03 00 00 11 83 82 01 84 00 00 00 01 a7
f9 5c 95 c7 9a a7 86 3f e4 8f 68 1a aa db 5c e9
ba 6e 6c 1e ff 79 54 4d 5a 6d b0 16 fe f4 9e 81
38 29 55 cb ac c2 b1 79 1b 01 95 14 aa da de d5
81 47 57 77 44 d7 e9 a4 44 23 b5 f3 fa c4 69 f8
14 9f 76 11 22 14 9c ce ba 68 90 bf 98 f1 99 25
9b a1 c4 bb 73 e5 92 c9 cc 42 01 e1 44 07 a4 95
```

Figure 3.10. Public key modulus

Figure 3.11, “Public key Barrett factor” The marked values from Figure 3.11, “Public key Barrett factor” depict the key's "Barrett factor", as extracted from the `.cer` file. Note the value `0` padding (leading bytes), which have to be stripped.

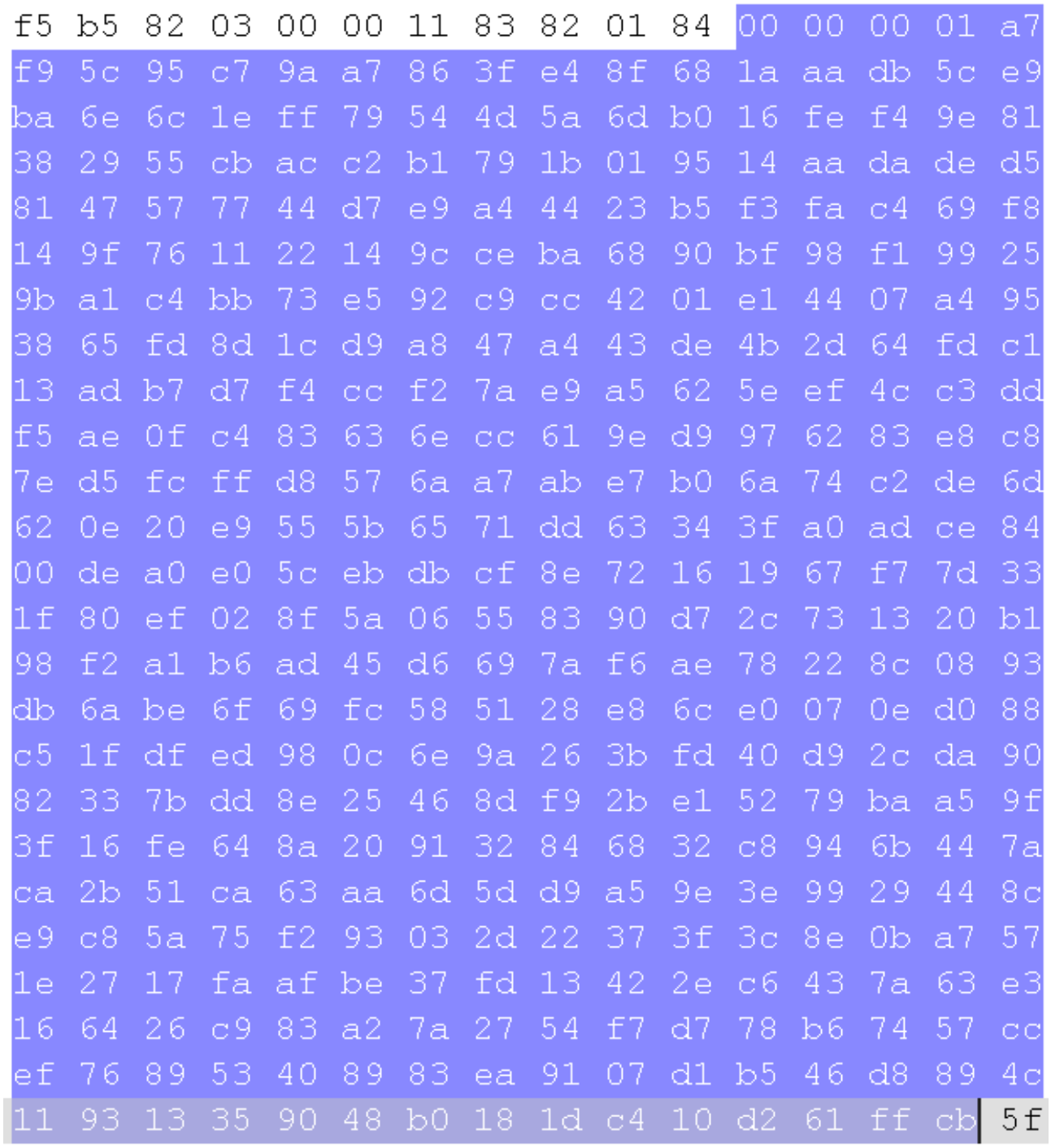

Figure 3.11. Public key Barrett factor

If you are using the asymmetrical key extract interfaces provided by the `Csm`, the `inputKey[]` array has to be set as shown in the following code snippet (the values correspond to Figure 3.9, “Public key modulus”, Figure 3.10, “Public key modulus” and Figure 3.11, “Public key Barrett factor”):

```
uint8 inputKey[] =
{
  /* Modulus */
  0x9Au, 0x93u, 0x53u, 0x65u, 0x65u, 0x5Fu, 0x1Cu, 0xD0u,
  ...,
  0x27u, 0xEDu, 0x65u, 0x69u, 0x5Du, 0x6Bu, 0xF5u, 0xB5u,

  /* Barrett factor */
  0x01u, 0xA7u, 0xF9u, 0x5Cu, 0x95u, 0xC7u, 0x9Au, 0xA7u,
  ...,
  0x18u, 0x1Du, 0xC4u, 0x10U, 0xD2u, 0x61u, 0xFFu, 0xCBu,

  /* Public exponent */
  0x00u, 0x00u, 0x00u, 0x00u, 0x00u, 0x00u, 0x00u, 0x00u,
  ...,
  0x00u, 0x00u, 0x00u, 0x00u, 0x00u, 0x00u, 0x00u, 0x11u
}
```

The following code snippet depicts the content of the `SignatureVerifyKeyPtr` (the values correspond to Figure 3.9, “Public key modulus”, Figure 3.10, “Public key modulus” and Figure 3.11, “Public key Barrett factor”):

```
const Csm_AsymPublicKeyType SignatureVerifyKeyPtr =
{
  0U,       /* Dummy length element for compatibility with Csm_AsymPublicKeyType */
  {
    /* Public exponent */
    0x00000001u,          /* Length of the public key exponent in words */
    0x00000011u,          /* Least significant bytes of the public exponent */
    0x00u,
    0x00u,
    ...,
    0x00u,                 /* Most significant bytes of the public exponent */

    /* Modulus */
    0x00000060u,           /* Length of the public key modulus in words */
    0x5D6BF5B5u,           /* Least significant bytes of the modulus */
    0x27ED6569u,
    0x4F0492E6u,
    ...,
    0x9A935365U,           /* Most significant bytes of the modulus */

    /* Barrett factor */
    0x00000061u,           /* Length of the public key Barrett factor in words */
    0xD261FFCBu,           /* Least significant bytes of the Barrett factor */
    0x181DC410u,
    0x359048B0u,
```

```
    ...,
    0x01A7F95Cu            /* Most significant bytes of the Barrett factor */
  }
}
```

3.5. Signature verify key for elliptic curve primitives

The following code snippet describes the format of the public key used by the EB crypto library and how to initialize the content of the data structure containing the public key. The length of the elliptic curve point is defined by the value of the `CryEllipticCurve` configuration parameter.

You have to call the `Csm_AsymPublicKeyExtractStart()`, `Csm_AsymPublicKeyExtractUpdate()` and `Csm_AsymPublicKeyExtractFinish()` APIs from integration code using the ID of the newly added Csm AsymPublicKeyExtract configuration.

The `Csm_AsymPublicKeyExtractFinish()` API expects an output pointer of type `Csm_AsymPublicKeyType` (compatible to `Cry`'s internal `Cry_EdDSAVrfyKeyType` type) to store the extracted key.

Public key example passed as input to Asymetric Public Key Extract primitive:

```
uint8 inputKey[] =
{
  /* Point */
  0x9Au, 0x93u, 0x53u, 0x65u, 0x65u, 0x5Fu, 0x1Cu, 0xD0u,
  ...,
  0x27u, 0xEDu, 0x65u, 0x69u, 0x5Du, 0x6Bu, 0xF5u, 0xB5u,
}
```

Encoded public key:

```
const Csm_AsymPublicKeyType SignatureVerifyKeyPtr =
{
  0U,       /* Dummy length element for compatibility with Csm_AsymPublicKeyType */

    /* Point */
    0x00000060u,           /* Length of the public key modulus in words */
    0x5D6BF5B5u,           /* Least significant bytes of the modulus */
    0x27ED6569u,
    0x4F0492E6u,
```

```
        ...,
        0x9A935365U,           /* Most significant bytes of the modulus */
      }
    }
```

4. Cry module references

4.1. Configuration parameters

Containers included		
Container name	**Multiplicity**	**Description**
CommonPublishedInformation	1..1	**Label:** Common Published Information Common container, aggregated by all modules. It contains published information about vendor and versions.
CryGeneral	1..1	The general configuration
CrySHA1	1..1	The configurations of SHA1
CrySHA2	1..1	The configurations of SHA2
CrySHA3	1..1	The configurations of SHA3
CryRsaSsaPssVerify	1..1	The configurations of Signature RsaSsa PSS Verify
CryRsaSsaV15Generate	1..1	The configurations of RsaSsa V15(PKCS1) Signature Generate
CryRsaSsaV15Verify	1..1	The configurations of Signature RsaSsa V15(PKCS1) Verify
CryEdDSAVrfy	1..1	The configurations of EdDSA verification with Ed25519ph curve
CryEdDSAGen	1..1	The configurations of ECDSA generation with Ed25519ph curve
CryMD5	1..1	The configurations of MD5
CryAESEncrypt	1..1	The configurations of AES encrypt
CryAESDecrypt	1..1	The configurations of AES decrypt
CryCMACVrfy	1..1	The configurations of CMAC
CryCMACGen	1..1	The configurations of CMAC
CrySSGGenerate	1..1	The configurations of the self shrinking generator
CrySSGSeed	1..1	The configurations of the self shrinking generator
CrySSGState	1..1	The configurations of the self shrinking generator
CryKeyExtractSym	1..1	The configurations of symmmetrical key extract
CryCVCPublicKeyExtract	1..1	**Label:** CVCPublicKeyExtract configurations The configurations of the CVC public key extract

Containers included		
CryCbcPkcs7Encrypt	1..1	The configurations of CBC encryption with PKCS#7 padding
CryCbcPkcs7Decrypt	1..1	The configurations of CBC decryption with PKCS#7 padding
CryCcmEncrypt	1..1	The configurations of Ccm encryption
CryCcmDecrypt	1..1	The configurations of Ccm decryption
CryHMACVrfy	1..1	The configurations of HMAC Verify
CryHMACGen	1..1	The configurations of HMAC Generate
CryExtractRsaPublicKey	1..1	**Label:** ExtractRsaPublicKey configurations The configurations of RSA public key extract
CryCrc	1..1	The configurations of cyclic redundancy check
CryExtractECPubKey	1..1	**Label:** ExtractECPubKey configurations The configurations of public elliptic key extract
PublishedInformation	1..1	**Label:** EB Published Information Additional published parameters not covered by Common-PublishedInformation container.

Parameters included	
Parameter name	**Multiplicity**
IMPLEMENTATION_CONFIG_VARIANT	1..1

Parameter Name	IMPLEMENTATION_CONFIG_VARIANT
Label	Config Variant
Description	Select the configuration variant. Currently only PreCompile is supported.
Multiplicity	1..1
Type	ENUMERATION
Default value	VariantPreCompile
Range	VariantPreCompile

4.1.1. CommonPublishedInformation

Parameters included	
Parameter name	**Multiplicity**
ArMajorVersion	1..1
ArMinorVersion	1..1

Parameters included	
ArPatchVersion	1..1
SwMajorVersion	1..1
SwMinorVersion	1..1
SwPatchVersion	1..1
ModuleId	1..1
VendorId	1..1
Release	1..1

Parameter Name	ArMajorVersion
Label	AUTOSAR Major Version
Description	Major version number of AUTOSAR specification on which the appropriate implementation is based on.
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	1
Configuration class	PublishedInformation:
Origin	Elektrobit Automotive GmbH

Parameter Name	ArMinorVersion
Label	AUTOSAR Minor Version
Description	Minor version number of AUTOSAR specification on which the appropriate implementation is based on.
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	2
Configuration class	PublishedInformation:
Origin	Elektrobit Automotive GmbH

Parameter Name	ArPatchVersion
Label	AUTOSAR Patch Version
Description	Patch level version number of AUTOSAR specification on which the appropriate implementation is based on.
Multiplicity	1..1
Type	INTEGER_LABEL

Default value	0
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

Parameter Name	**SwMajorVersion**
Label	Software Major Version
Description	Major version number of the vendor specific implementation of the module.
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	2
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

Parameter Name	**SwMinorVersion**
Label	Software Minor Version
Description	Minor version number of the vendor specific implementation of the module. The numbering is vendor specific.
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	6
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

Parameter Name	**SwPatchVersion**
Label	Software Patch Version
Description	Patch level version number of the vendor specific implementation of the module. The numbering is vendor specific.
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	6
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

Parameter Name	**ModuleId**
Label	Numeric Module ID

Description	Module ID of this module from Module List
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	0
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

Parameter Name	**VendorId**
Label	Vendor ID
Description	Vendor ID of the dedicated implementation of this module according to the AUTOSAR vendor list
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	1
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

Parameter Name	**Release**
Label	Release Information
Multiplicity	1..1
Type	STRING_LABEL
Default value	
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

4.1.2. CryGeneral

Parameters included	
Parameter name	**Multiplicity**
CryLNAlgorithmImplementation	1..1
CryInterruptableLN	1..1
CryLNPlatformDoubleWordSupport	1..1
CrySHAOneAndTwoImplementation	1..1

Parameters included	
CryCrcImplementationVariant	1..1

Parameter Name	CryLNAlgorithmImplementation
Label	LN algorithm implementation variant
Description	Selects the implementation variant of long number library. Range: ► CRY_LN_MEMORY_OPTIMIZED This implementation variant offers a trade-off between RAM usage and processing time. ► CRY_LN_SPEED_OPTIMIZED This implementation variant allocates RAM on the stack during the execution of long number functions to achieve a shorter processing time.
Multiplicity	1..1
Type	ENUMERATION
Default value	CRY_LN_MEMORY_OPTIMIZED
Range	CRY_LN_MEMORY_OPTIMIZED CRY_LN_SPEED_OPTIMIZED

Parameter Name	CryInterruptableLN
Label	Interruptable LN operations
Description	If disabled, the Cry module will process all operations related to long numbers in a single main function call. This implementation variant trades CPU usage per main function call for achieving a smaller total processing time of the algorithm. If enabled, the Cry module splits all operations related to long numbers across several main function calls. This implementation variant offers a trade-off between the total processing time of the algorithm and CPU usage per main function call. Note: this configuration parameter is only available if the "memory optimized" long number library implementation variant is used (CryLNAlgorithmImplementation = CRY_LN_MEMORY_OPTIMIZED).
Multiplicity	1..1
Type	BOOLEAN
Default value	true

Parameter Name	CryLNPlatformDoubleWordSupport
Label	Support for double words
Description	Switch to enable/disable support for double words (words having the length equal to double the platform size). ► FALSE: The Cry module uses single words for calculating partial results during long number operations by splitting the operands in lower/upper parts. ► TRUE: The Cry module uses double words for calculating partial results during long number operations. Note: this configuration parameter is only available if the "speed optimized" long number library implementation variant is used (CryLNAlgorithmImplementation = CRY_LN_SPEED_OPTIMIZED).
Multiplicity	1..1
Type	BOOLEAN
Default value	false

Parameter Name	CrySHAOneAndTwoImplementation
Label	SHA-1/SHA-2 implementation variant
Description	Selects the implementation variant of the SHA-1 and SHA-2 hashing algorithms. Range: ► CRY_SHAONEANDTWO_INTERRUPTABLE The Cry module splits SHA-1/SHA-2 hash computations across several main function calls. This implementation variant offers a trade-off between the total processing time of the algorithm and CPU usage per main function call. ► CRY_SHAONEANDTWO_FAST The Cry module performs as many SHA-1/SHA-2 hash computations as possible per main function call. This implementation variant trades CPU usage per main function call for achieving a smaller total processing time of the algorithm.
Multiplicity	1..1
Type	ENUMERATION
Default value	CRY_SHAONEANDTWO_INTERRUPTABLE
Range	CRY_SHAONEANDTWO_INTERRUPTABLE

	CRY_SHAONEANDTWO_FAST

Parameter Name	CryCrcImplementationVariant
Label	CRC algorithm implementation variant
Description	Selects the implementation variant of crcTable.
Multiplicity	1..1
Type	ENUMERATION
Default value	CRY_CRC_MEMORY_OPTIMIZED
Range	CRY_CRC_SPEED_OPTIMIZED CRY_CRC_MEMORY_OPTIMIZED

4.1.3. CrySHA1

Containers included		
Container name	**Multiplicity**	**Description**
CrySHA1Config	0..32	

Parameters included	
Parameter name	**Multiplicity**
CrySHA1ImmediateRestartEnabled	1..1

Parameter Name	CrySHA1ImmediateRestartEnabled
Label	Enable the cancelation of ongoing requests
Description	Enable the cancelation of an ongoing calculation regardless of the configuration ID.
Multiplicity	1..1
Type	BOOLEAN
Default value	false

4.1.4. CrySHA1Config

Parameters included	
Parameter name	**Multiplicity**

Parameters included	
CrySHA1Type	1..1
CrySHA1IterationsPerMain	1..1

Parameter Name	CrySHA1Type
Label	Prime
Description	Prime used
Multiplicity	1..1
Type	ENUMERATION
Default value	CRY_SHA_1
Range	CRY_SHA_1

Parameter Name	CrySHA1IterationsPerMain
Label	Number of iterations per MainFunction()
Description	Defines the number of SHA1 algoritm iterations per Cry_SHA1MainFunction() call during the compression phase. This allows to fine tune the execution time per main function call and thus the number of calls. The value 0 means that the number of iterations shall not be limited. Note: This configuration parameter is available only if SHA-1/SHA-2 interruptable code variant is used (`CrySHAOneAndTwoImplementation` = `CRY_SHAONEANDTWO_INTERRUPTABLE`)
Multiplicity	1..1
Type	INTEGER
Default value	1
Range	<=65535 >=0
Configuration class	**PreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

4.1.5. CrySHA2

Containers included		
Container name	**Multiplicity**	**Description**

Containers included		
CrySHA2Config	0..32	

Parameters included	
Parameter name	**Multiplicity**
CrySHA2ImmediateRestartEnabled	1..1

Parameter Name	CrySHA2ImmediateRestartEnabled
Label	Enable the cancelation of ongoing requests
Description	Enable the cancelation of an ongoing calculation regardless of the configuration ID.
Multiplicity	1..1
Type	BOOLEAN
Default value	false

4.1.6. CrySHA2Config

Parameters included	
Parameter name	**Multiplicity**
CrySHA2Type	1..1
CrySHA2IterationsPerMain	1..1
CrySHA2SupportRestart	1..1

Parameter Name	CrySHA2Type
Label	Prime
Description	Prime used
Multiplicity	1..1
Type	ENUMERATION
Default value	CRY_SHA_256
Range	CRY_SHA_224
	CRY_SHA_256
	CRY_SHA_384
	CRY_SHA_512

Parameter Name	CrySHA2IterationsPerMain	
Label	Number of iterations per MainFunction()	
Description	Defines the number of SHA-2 algoritm iterations per Cry_SHA2MainFunction() call during the compression phase. This allows to fine tune the execution time per main function call and thus the number of calls. The value 0 means that the number of iterations shall not be limited. Note: This configuration parameter is available only if SHA-1/SHA-2 interruptable code variant is used (`CrySHAOneAndTwoImplementation` = `CRY_SHAONEANDTWO_INTERRUPTABLE`)	
Multiplicity	1..1	
Type	INTEGER	
Default value	1	
Range	<=65535 >=0	
Configuration class	**PreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	CrySHA2SupportRestart
Label	Enable the cancelation of ongoing requests
Description	If enabled, the restart feature of the service is enabled. **Important:** If Cry_SHA2Start can interrupt Cry_SHA2MainFunction, a service should only be restarted after the last expected callback notification has been called. Otherwise, callback notifications related to the canceled calculation might be called. Please ensure that the corresponding parameter CsmHashEnableRestart in the CSM has the same value as this parameter.
Multiplicity	1..1
Type	BOOLEAN
Default value	false

4.1.7. CrySHA3

Containers included		
Container name	**Multiplicity**	**Description**

Containers included		
CrySHA3Config	0..32	

Parameters included	
Parameter name	**Multiplicity**
CrySHA3ImmediateRestartEnabled	1..1

Parameter Name	CrySHA3ImmediateRestartEnabled
Label	Enable the cancelation of ongoing requests
Description	Enable the cancelation of an ongoing calculation regardless of the configuration ID.
Multiplicity	1..1
Type	BOOLEAN
Default value	false

4.1.8. CrySHA3Config

Parameters included	
Parameter name	**Multiplicity**
CrySHA3Type	1..1
CrySHA3SupportRestart	1..1

Parameter Name	CrySHA3Type
Label	Prime
Description	Prime used
Multiplicity	1..1
Type	ENUMERATION
Default value	CRY_SHA3_256
Range	CRY_SHA3_224
	CRY_SHA3_256
	CRY_SHA3_384
	CRY_SHA3_512

Parameter Name	CrySHA3SupportRestart

Label	Enable the cancelation of ongoing requests
Description	If enabled, the restart feature of the service is enabled. **Important:** If Cry_SHA3Start can interrupt Cry_SHA3MainFunction, a service should only be restarted after the last expected callback notification has been called. Otherwise, callback notifications related to the canceled calculation might be called. Please ensure that the corresponding parameter CsmHashEnableRestart in the CSM has the same value as this parameter.
Multiplicity	1..1
Type	BOOLEAN
Default value	false

4.1.9. CryRsaSsaPssVerify

Containers included		
Container name	**Multiplicity**	**Description**
CryRsaSsaPssVerifyConfig	0..32	

Parameters included	
Parameter name	**Multiplicity**
CryRsaSsaPssVerifyImmediateRestartEnabled	1..1
CryRsaSsaPssVerifyUseTimeSlices	1..1
CryRsaSsaPssVerifyNumberOfTimeSlices	1..1
CryRsaSsaPssVerifyUseCbk	1..1
CryRsaSsaPssTimeSlicesCbk	1..1

Parameter Name	**CryRsaSsaPssVerifyImmediateRestartEnabled**
Label	Enable the cancelation of ongoing requests
Description	Enable the cancelation of an ongoing calculation regardless of the configuration ID.
Multiplicity	1..1
Type	BOOLEAN
Default value	false

Parameter Name	**CryRsaSsaPssVerifyUseTimeSlices**

Label	Use time slicing for RSASSA-PSS signature verification.
Description	If enabled, the signature verification is split into time slices. Specifically, the modular exponentiation performed during signature verification is split into several steps.
Multiplicity	1..1
Type	BOOLEAN
Default value	false

Parameter Name	**CryRsaSsaPssVerifyNumberOfTimeSlices**
Label	Number of RsaSsaPss time slices
Description	The maximum number of steps of the modular exponentiation performed in one Cry main function call during the RSASSA-PSS signature verification. NOTE: This configuration parameter is disabled, when CryRsaSsaPssVerifyUseCbk configuration parameter is enabled.
Multiplicity	1..1
Type	INTEGER
Default value	1
Range	>=1 \<=4294967295

Parameter Name	**CryRsaSsaPssVerifyUseCbk**
Label	Use configured callback function which returns maximum number of time slices
Description	If enabled, the configuration parameter CryRsaSsaPssTimeSlicesCbk is enabled, to configure the name of a callback which returns the maximum number of time slices to be used during RSASSA-PSS signature verification.
Multiplicity	1..1
Type	BOOLEAN
Default value	false

Parameter Name	**CryRsaSsaPssTimeSlicesCbk**
Label	CryRsaSsaPssTimeSlicesCbk
Description	Optional configuration parameter which can be used to configure the name of the callback which returns the maximum number of time slices to be used during RSASSA-PSS signature verification. NOTE: This configuration parameter is enabled, only when CryRsaSsaPssVerifyUseCbk configuration parameter is enabled.

Multiplicity	1..1	
Type	FUNCTION-NAME	
Configuration class	**PreCompile:**	VariantPreCompile

4.1.10. CryRsaSsaPssVerifyConfig

Parameters included	
Parameter name	**Multiplicity**
CryRsaSsaPssVerifyHashCfgRef	1..1
CryRsaSsaPssVerifyKeyLength	1..1
CryRsaSsaPssVerifySaltLength	1..1
CryRsaSsaPssVerifyB64Encoded	1..1
CryRsaSsaPssVerifyUseBarrett	1..1
CryRsaSsaPssVerifySupportRestart	1..1

Parameter Name	**CryRsaSsaPssVerifyHashCfgRef**
Label	Hash configuration
Description	Hash configuration used
Multiplicity	1..1
Type	REFERENCE

Parameter Name	**CryRsaSsaPssVerifyKeyLength**
Label	Key Length
Description	Key Length in bytes used in the signature verification
Multiplicity	1..1
Type	INTEGER
Default value	384

Parameter Name	**CryRsaSsaPssVerifySaltLength**
Label	Salt Length
Description	Salt Length in bytes used in the signature verification
Multiplicity	1..1
Type	INTEGER
Default value	384

Range	>=0 \<=4294967295

Parameter Name	**CryRsaSsaPssVerifyB64Encoded**
Label	Base64 Encoded
Description	The signature is Base64-encoded.
Multiplicity	1..1
Type	BOOLEAN
Default value	false

Parameter Name	**CryRsaSsaPssVerifyUseBarrett**
Label	Barrett reduction
Description	If enabled, the Barret reduction is used to verify the signature.
Multiplicity	1..1
Type	BOOLEAN
Default value	false

Parameter Name	**CryRsaSsaPssVerifySupportRestart**
Label	Enable the cancelation of ongoing requests
Description	If enabled, the restart feature of the service is enabled. **Important:** If Cry_RsaSsaPssVerifyStart can interrupt Cry_RsaSsaPssVerify-MainFunction, a service should only be restarted after the last expected callback notification has been called. Otherwise, callback notifications related to the canceled calculation might be called. Please ensure that the corresponding parameter CsmSignatureVerifyEnableRestart in the CSM has the same value as this parameter.
Multiplicity	1..1
Type	BOOLEAN
Default value	false

4.1.11. CryRsaSsaV15Generate

Containers included		
Container name	**Multiplicity**	**Description**
CryRsaSsaV15GenerateConfig	0..32	

4.1.12. CryRsaSsaV15GenerateConfig

Parameters included	
Parameter name	**Multiplicity**
CryRsaSsaV15GenerateHashCfgRef	1..1
CryRsaSsaV15GenerateKeyLength	1..1

Parameter Name	CryRsaSsaV15GenerateHashCfgRef
Label	Hash configuration
Description	Hash configuration used
Multiplicity	1..1
Type	REFERENCE

Parameter Name	CryRsaSsaV15GenerateKeyLength
Label	Key Length
Description	Key length in bytes used in the signature generation
Multiplicity	1..1
Type	INTEGER
Default value	384

4.1.13. CryRsaSsaV15Verify

Containers included		
Container name	**Multiplicity**	**Description**
CryRsaSsaV15VerifyConfig	0..32	

Parameters included	
Parameter name	**Multiplicity**
CryRsaSsaV15VerifyImmediateRestartEnabled	1..1
CryRsaSsaV15VerifyUseTimeSlices	1..1
CryRsaSsaV15VerifyNumberOfTimeSlices	1..1
CryRsaSsaV15VerifyUseCbk	1..1
CryRsaSsaV15TimeSlicesCbk	1..1

Parameter Name	CryRsaSsaV15VerifyImmediateRestartEnabled
Label	Enable the cancelation of ongoing requests
Description	Enable the cancelation of an ongoing calculation regardless of the configuration ID.
Multiplicity	1..1
Type	BOOLEAN
Default value	false

Parameter Name	CryRsaSsaV15VerifyUseTimeSlices
Label	Use time slices for RSASSA-PKCS1-v1_5 signature verification.
Description	If enabled, the signature verification is split into time slices. Specifically, the modular exponentiation performed during RSASSA-PKCS1-v1_5 signature verification is split into several steps.
Multiplicity	1..1
Type	BOOLEAN
Default value	false

Parameter Name	CryRsaSsaV15VerifyNumberOfTimeSlices
Label	Number of RsaSsaV15 time slices
Description	The maximum number of steps of the modular exponentiation performed in one Cry main function call during the RSASSA-PKCS1-v1_5 signature verification. NOTE: This configuration parameter is disabled, when CryRsaSsaV15VerifyUseCbk configuration parameter is enabled.
Multiplicity	1..1
Type	INTEGER
Default value	1
Range	>=1 \<=4294967295

Parameter Name	CryRsaSsaV15VerifyUseCbk
Label	Use configured callback function which returns maximum number of time slices.
Description	If enabled, the configuration parameter CryRsaSsaV15TimeSlicesCbk is enabled, to configure the name of a callback which returns the maximum number of time slices to be used during RSASSA-PKCS1-v1_5 signature verification.
Multiplicity	1..1

Type	BOOLEAN
Default value	false

Parameter Name	**CryRsaSsaV15TimeSlicesCbk**	
Label	CryRsaSsaV15TimeSlicesCbk	
Description	Optional configuration parameter which can be used to configure the name of the callback which returns the maximum number of time slices to be used during RSASSA-PKCS1-v1_5 signature verification. NOTE: This configuration parameter is enabled, only when CryRsaSsaV15VerifyUseCbk configuration parameter is enabled.	
Multiplicity	1..1	
Type	FUNCTION-NAME	
Configuration class	**PreCompile:**	VariantPreCompile

4.1.14. CryRsaSsaV15VerifyConfig

Parameters included	
Parameter name	**Multiplicity**
CryRsaSsaV15VerifyHashCfgRef	1..1
CryRsaSsaV15VerifyKeyLength	1..1
CryRsaSsaV15VerifyB64Encoded	1..1
CryRsaSsaV15VerifyUseBarrett	1..1
CryRsaSsaV15VerifySupportRestart	1..1

Parameter Name	**CryRsaSsaV15VerifyHashCfgRef**
Label	Hash configuration
Description	Hash configuration used
Multiplicity	1..1
Type	REFERENCE

Parameter Name	**CryRsaSsaV15VerifyKeyLength**
Label	Key Length
Description	Key Length in bytes used in the signature verification
Multiplicity	1..1

Type	INTEGER
Default value	384

Parameter Name	**CryRsaSsaV15VerifyB64Encoded**
Label	Base64 Encoded
Description	The signature is Base64-encoded.
Multiplicity	1..1
Type	BOOLEAN
Default value	false

Parameter Name	**CryRsaSsaV15VerifyUseBarrett**
Label	Barrett reduction
Description	If enabled, the Barret reduction is used to verify the signature.
Multiplicity	1..1
Type	BOOLEAN
Default value	false

Parameter Name	**CryRsaSsaV15VerifySupportRestart**
Label	Enable the cancelation of ongoing requests
Description	If enabled, the restart feature of the service is enabled. **Important:** If Cry_RsaSsaV15VerifyStart can interrupt Cry_-RsaSsaV15VerifyMainFunction, a service should only be restarted after the last expected callback notification has been called. Otherwise, callback notifications related to the canceled calculation might be called. Please ensure that the corresponding parameter CsmSignatureVerifyEn-ableRestart in the CSM has the same value as this parameter.
Multiplicity	1..1
Type	BOOLEAN
Default value	false

4.1.15. CryEdDSAVrfy

Containers included		
Container name	**Multiplicity**	**Description**

Containers included		
CryEdDSAVrfyConfig	0..32	

Parameters included	
Parameter name	**Multiplicity**
CryEdDSAVrfyImmediateRestartEnabled	1..1

Parameter Name	CryEdDSAVrfyImmediateRestartEnabled
Label	Enable the cancelation of ongoing requests
Description	Enable the cancelation of an ongoing calculation regardless of the configuration ID.
Multiplicity	1..1
Type	BOOLEAN
Default value	false

4.1.16. CryEdDSAVrfyConfig

Parameters included	
Parameter name	**Multiplicity**
CryEdDSAVrfyHashCfgRef	1..1
CryEdDSAVrfySupportRestart	1..1
CryEdDSAVrfyNumberOfTimeSlices	1..1

Parameter Name	CryEdDSAVrfyHashCfgRef
Label	Hash configuration
Description	Hash configuration used A reference to SHA-512 is mandatory.
Multiplicity	1..1
Type	REFERENCE

Parameter Name	CryEdDSAVrfySupportRestart
Label	Enable the cancelation of ongoing requests
Description	If enabled, the restart feature of the service is enabled. **Important:** If Cry_EdDSAVrfyStart can interrupt Cry_EdDSAVrfyMainFunction, a service should only be restarted after the last expected callback notification has

	been called. Otherwise, callback notifications related to the canceled calculation might be called. Please ensure that the corresponding parameter CsmHashEnableRestart in the CSM has the same value as this parameter.
Multiplicity	1..1
Type	BOOLEAN
Default value	false

Parameter Name	**CryEdDSAVrfyNumberOfTimeSlices**
Label	Number of time slices for EdDSA signature verification
Description	The maximum number of steps of the point operations performed in one Cry main function call during the EdDSA signature verification. Note: The value of this configuration parameter allows the integrator of the Cry module to achieve a balance between the total processing time of the algorithm and the CPU usage per main function call. ► `0`: The Cry module does not limit the number of time slices per `Cry_EdDSAVrfyMainFunction()` function call. ► `1`: The Cry module processes a single time slice (i.e. the slicing mechanism is disabled) per `Cry_EdDSAVrfyMainFunction()` function call. ► `> 1`: The Cry module processes the configured number of slices `Cry_EdDSAVrfyMainFunction()` function call.
Multiplicity	1..1
Type	INTEGER
Default value	1
Range	>=0 --- <=65535

4.1.17. CryEdDSAGen

Containers included		
Container name	**Multiplicity**	**Description**
CryEdDSAGenConfig	0..32	

Parameters included	
Parameter name	**Multiplicity**

Parameters included	
CryEdDSAGenImmediateRestartEnabled	1..1

Parameter Name	CryEdDSAGenImmediateRestartEnabled
Label	Enable the cancelation of ongoing requests
Description	Enable the cancelation of an ongoing calculation regardless of the configuration ID.
Multiplicity	1..1
Type	BOOLEAN
Default value	false

4.1.18. CryEdDSAGenConfig

Parameters included	
Parameter name	Multiplicity
CryEdDSAGenHashCfgRef	1..1
CryEdDSAGenSupportRestart	1..1
CryEdDSAGenNumberOfTimeSlices	1..1

Parameter Name	CryEdDSAGenHashCfgRef
Label	Hash configuration
Description	Hash configuration used A reference to SHA-512 is mandatory.
Multiplicity	1..1
Type	REFERENCE

Parameter Name	CryEdDSAGenSupportRestart
Label	Enable the cancelation of ongoing requests
Description	If enabled, the restart feature of the service is enabled. **Important:** If Cry_EdDSAGenStart can interrupt Cry_EdDSAGenMainFunction, a service should only be restarted after the last expected callback notification has been called. Otherwise, callback notifications related to the canceled calculation might be called. Please ensure that the corresponding parameter CsmHashEnableRestart in the CSM has the same value as this parameter.

Multiplicity	1..1
Type	BOOLEAN
Default value	false

Parameter Name	**CryEdDSAGenNumberOfTimeSlices**
Label	Number of time slices for EdDSA signature generation
Description	The maximum number of steps of the point operations performed in one Cry main function call during the EdDSA signature generation. Note: The value of this configuration parameter allows the integrator of the Cry module to achieve a balance between the total processing time of the algorithm and the CPU usage per main function call. ► `0`: The Cry module does not limit the number of time slices per `Cry_EdDSAGenMainFunction()` function call. ► `1`: The Cry module processes a single time slice (i.e. the slicing mechanism is disabled) per `Cry_EdDSAGenMainFunction()` function call. ► `> 1`: The Cry module processes the configured number of slices `Cry_EdDSAGenMainFunction()` function call.
Multiplicity	1..1
Type	INTEGER
Default value	1
Range	>=0 --- <=65535

4.1.19. CryMD5

Containers included		
Container name	**Multiplicity**	**Description**
CryMD5Config	0..32	

Parameters included	
Parameter name	**Multiplicity**
CryMD5ImmediateRestartEnabled	1..1

Parameter Name	**CryMD5ImmediateRestartEnabled**

Label	Enable the cancelation of ongoing requests
Description	Enable the cancelation of an ongoing calculation regardless of the configuration ID.
Multiplicity	1..1
Type	BOOLEAN
Default value	false

4.1.20. CryMD5Config

4.1.21. CryAESEncrypt

Containers included		
Container name	**Multiplicity**	**Description**
CryAESEncryptConfig	0..32	

Parameters included	
Parameter name	**Multiplicity**
PZ1ACryOptimEncry	1..1
CryAESEncryptImplementation	0..1

Parameter Name	**PZ1ACryOptimEncry**
Label	PZ1A Cry Optimization
Description	Enables the possibility for interruption of the main function execution for the AES Encrypt algorithm. If enabled, the execution of the main function is divided in small execution steps, allowing to have OS interruptions between them. As a consequence the total time execution of a Cry job will take longer, as it can be interupted by other running processes/tasks. This is useful when processing large amount of data with Cry (which takes a long time), enabling non-blocking operation of other processes/tasks.
Multiplicity	1..1
Type	BOOLEAN

Default value	false

Parameter Name	**CryAESEncryptImplementation**	
Label	Cry AES Encrypt Implementation	
Description	Selects the implementation variant of the AES encryption algorithm. Range: ► CRY_AES_ENCRYPT_RESOURCE_EFFICIENT This implementation variant offers a trade-off between RAM usage the number of CPU instructions. Requires 1279 bytes for look-up tables, which are shared between encryption and decryption. ► CRY_AES_ENCRYPT_FAST This implementation variant trades the number of CPU instructions for RAM usage. Requires 1064 bytes for look-up tables only for the encryption part and an additional 512 bytes for tables shared between encryption and decryption.	
Multiplicity	0..1	
Type	ENUMERATION	
Default value	CRY_AES_ENCRYPT_RESOURCE_EFFICIENT	
Range	CRY_AES_ENCRYPT_RESOURCE_EFFICIENT	
	CRY_AES_ENCRYPT_FAST	
Configuration class	**VariantPreCompile:**	VariantPreCompile

4.1.22. CryAESEncryptConfig

Parameters included	
Parameter name	**Multiplicity**
CryAESEncryptType	1..1

Parameter Name	**CryAESEncryptType**
Label	Key Length
Description	Key Length used

Multiplicity	1..1
Type	ENUMERATION
Default value	CRY_AES_KEY_256
Range	CRY_AES_KEY_128 CRY_AES_KEY_192 CRY_AES_KEY_256

4.1.23. CryAESDecrypt

Containers included		
Container name	**Multiplicity**	**Description**
CryAESDecryptConfig	0..32	

Parameters included	
Parameter name	**Multiplicity**
PZ1ACryOptimDecry	1..1
CryAESDecryptImplementation	0..1

Parameter Name	**PZ1ACryOptimDecry**
Label	PZ1A Cry Optimization
Description	Enables the possibility for interruption of the main function execution for the AES Decrypt algorithm. If enabled, the execution of the main function is divided in small execution steps, allowing to have OS interruptions between them. As a consequence the total time execution of a Cry job will take longer, as it can be interupted by other running processes/tasks. This is useful when processing large amount of data with Cry (which takes a long time), enabling non-blocking operation of other processes/tasks.
Multiplicity	1..1
Type	BOOLEAN
Default value	false

Parameter Name	**CryAESDecryptImplementation**

Label	Cry AES Decrypt Implementation
Description	Selects the implementation variant of the AES decryption algorithm. Range: ► `CRY_AES_DECRYPT_RESOURCE_EFFICIENT` This implementation variant offers a trade-off between RAM usage the number of CPU instructions. Requires 1279 bytes for look-up tables, which are shared between decryption and encryption. ► `CRY_AES_DECRYPT_FAST` This implementation variant trades the number of CPU instructions for RAM usage. Requires 1064 bytes for look-up tables only for the decryption part and an additional 512 bytes for tables shared between decryption and encryption.
Multiplicity	0..1
Type	ENUMERATION
Default value	CRY_AES_DECRYPT_RESOURCE_EFFICIENT
Range	CRY_AES_DECRYPT_RESOURCE_EFFICIENT CRY_AES_DECRYPT_FAST
Configuration class	**VariantPreCompile:** VariantPreCompile

4.1.24. CryAESDecryptConfig

Parameters included	
Parameter name	**Multiplicity**
CryAESDecryptType	1..1

Parameter Name	**CryAESDecryptType**
Label	Key Length
Description	Key Length used
Multiplicity	1..1
Type	ENUMERATION

Default value	CRY_AES_KEY_256
Range	CRY_AES_KEY_128 CRY_AES_KEY_192 CRY_AES_KEY_256

4.1.25. CryCMACVrfy

Containers included		
Container name	**Multiplicity**	**Description**
CryCMACVrfyConfig	0..32	

4.1.26. CryCMACVrfyConfig

Parameters included	
Parameter name	**Multiplicity**
CryCMACVrfySymBlockEncryptCfgRef	1..1
CryCMACVrfyType	1..1

Parameter Name	**CryCMACVrfySymBlockEncryptCfgRef**
Label	SymBlockEncrypt configuration
Description	SymBlockEncrypt configuration used
Multiplicity	1..1
Type	REFERENCE

Parameter Name	**CryCMACVrfyType**
Label	MAC Type
Description	Type of One-Key CBC MAC: CMAC which is equivalent with OMAC1. OMAC2
Multiplicity	1..1
Type	ENUMERATION

Default value	CMAC
Range	CMAC OMAC2

4.1.27. CryCMACGen

Containers included		
Container name	**Multiplicity**	**Description**
CryCMACGenConfig	0..32	

4.1.28. CryCMACGenConfig

Parameters included	
Parameter name	**Multiplicity**
CryCMACGenSymBlockEncryptCfgRef	1..1
CryCMACGenType	1..1

Parameter Name	**CryCMACGenSymBlockEncryptCfgRef**
Label	SymBlockEncrypt configuration
Description	SymBlockEncrypt configuration used
Multiplicity	1..1
Type	REFERENCE

Parameter Name	**CryCMACGenType**
Label	MAC Type
Description	Type of One-Key CBC MAC: CMAC which is equivalent with OMAC1. OMAC2
Multiplicity	1..1
Type	ENUMERATION
Default value	CMAC

Range	CMAC
	OMAC2

4.1.29. CrySSGGenerate

Containers included		
Container name	**Multiplicity**	**Description**
CrySSGGenerateConfig	0..32	

Parameters included	
Parameter name	**Multiplicity**
CrySSGGenerateImmediateRestartEnabled	1..1

Parameter Name	**CrySSGGenerateImmediateRestartEnabled**
Label	Enable the cancelation of ongoing requests
Description	Enable the cancelation of an ongoing calculation regardless of the configuration ID.
Multiplicity	1..1
Type	BOOLEAN
Default value	false

4.1.30. CrySSGGenerateConfig

Parameters included	
Parameter name	**Multiplicity**
CrySSGRandomStateRef	1..1

Parameter Name	**CrySSGRandomStateRef**
Label	Random state
Description	Random state used
Multiplicity	1..1
Type	REFERENCE

4.1.31. CrySSGSeed

Containers included		
Container name	**Multiplicity**	**Description**
CrySSGSeedConfig	0..32	

Parameters included	
Parameter name	**Multiplicity**
CrySSGSeedImmediateRestartEnabled	1..1

Parameter Name	**CrySSGSeedImmediateRestartEnabled**
Label	Enable the cancelation of ongoing requests
Description	Enable the cancelation of an ongoing calculation regardless of the configuration ID.
Multiplicity	1..1
Type	BOOLEAN
Default value	false

4.1.32. CrySSGSeedConfig

Parameters included	
Parameter name	**Multiplicity**
CrySSGRandomStateRef	1..1

Parameter Name	**CrySSGRandomStateRef**
Label	Random state
Description	Random state used
Multiplicity	1..1
Type	REFERENCE

4.1.33. CrySSGState

Containers included		
Container name	**Multiplicity**	**Description**

Containers included		
CrySSGStateConfig	0..32	

4.1.34. CrySSGStateConfig

Parameters included	
Parameter name	**Multiplicity**
CrySSGStateSize	1..1

Parameter Name	CrySSGStateSize
Label	Random state size in bytes
Description	The size in bytes of the LFSR used during SSG Random Generate.
Multiplicity	1..1
Type	INTEGER
Default value	1
Range	>=1 <=20

4.1.35. CryKeyExtractSym

Containers included		
Container name	**Multiplicity**	**Description**
CryKeyExtractSymConfig	0..8	

Parameters included	
Parameter name	**Multiplicity**
CryKeyExtractSymImmediateRestartEnabled	1..1

Parameter Name	CryKeyExtractSymImmediateRestartEnabled
Label	Enable the cancelation of ongoing requests
Description	Enable the cancelation of an ongoing calculation regardless of the configuration ID.

Multiplicity	1..1
Type	BOOLEAN
Default value	false

4.1.36. CryKeyExtractSymConfig

Parameters included	
Parameter name	**Multiplicity**
CryKeyExtractSymType	1..1

Parameter Name	**CryKeyExtractSymType**
Label	Key Length
Description	Key Length used
Multiplicity	1..1
Type	ENUMERATION
Default value	CRY_KEY_SYM_256
Range	CRY_KEY_SYM_128
	CRY_KEY_SYM_192
	CRY_KEY_SYM_256

4.1.37. CryCVCPublicKeyExtract

Containers included		
Container name	**Multiplicity**	**Description**
CryCVCPublicKeyExtract-Config	0..32	

4.1.38. CryCVCPublicKeyExtractConfig

Parameters included	
Parameter name	**Multiplicity**

Parameters included	
CryCVCPublicKeyExtractCfgRef	1..1
CryExtractCVCRootCertificateLength	1..1
CryExtractCVCSigCertificateLength	1..1
CryCVCPublicKeyExtractLength	1..1
CryCVCPublicKeyExtractEnableCertChain	1..1

Parameter Name	CryCVCPublicKeyExtractCfgRef
Label	Signature verify configuration
Description	Signature verify configuration which is used
Multiplicity	1..1
Type	REFERENCE

Parameter Name	CryExtractCVCRootCertificateLength
Label	Length of the issuing certificate (in bytes)
Description	Length of the issuing certificate (in bytes) which is used
Multiplicity	1..1
Type	INTEGER
Default value	1
Range	>=1 \<=4294967295

Parameter Name	CryExtractCVCSigCertificateLength
Label	Length of the key certificate (in bytes)
Description	Length of the key certificate (in bytes) which is used
Multiplicity	1..1
Type	INTEGER
Default value	1
Range	>=0 \<=4294967295

Parameter Name	CryCVCPublicKeyExtractLength
Label	Key Length (in bytes)
Description	Key Length used

Multiplicity	1..1
Type	INTEGER
Default value	1

Parameter Name	**CryCVCPublicKeyExtractEnableCertChain**
Label	Use certificate chain verification
Description	This configuration parameter enables or disables the verification of the certificate chain. True: The key certificate is verified (i.e. its dates and signature) against the issuing certificate. False: The key is extracted from the key certificate without verifying the certificate against its issuing certificate
Multiplicity	1..1
Type	BOOLEAN
Default value	true

4.1.39. CryCbcPkcs7Encrypt

Containers included		
Container name	**Multiplicity**	**Description**
CryCbcPkcs7EncryptConfig	0..32	

4.1.40. CryCbcPkcs7EncryptConfig

Parameters included	
Parameter name	**Multiplicity**
CryCbcPkcs7EncryptSymBlockEncryptCfgRef	1..1

Parameter Name	**CryCbcPkcs7EncryptSymBlockEncryptCfgRef**
Label	SymBlockEncrypt configuration
Description	Symmetrical block encryption configuration used
Multiplicity	1..1
Type	REFERENCE

4.1.41. CryCbcPkcs7Decrypt

Containers included		
Container name	**Multiplicity**	**Description**
CryCbcPkcs7DecryptConfig	0..32	

4.1.42. CryCbcPkcs7DecryptConfig

Parameters included	
Parameter name	**Multiplicity**
CryCbcPkcs7DecryptSymBlockDecryptCfgRef	1..1

Parameter Name	CryCbcPkcs7DecryptSymBlockDecryptCfgRef
Label	SymBlockDecrypt configuration
Description	Symmetrical block decryption configuration used
Multiplicity	1..1
Type	REFERENCE

4.1.43. CryCcmEncrypt

Containers included		
Container name	**Multiplicity**	**Description**
CryCcmEncryptConfig	0..32	

4.1.44. CryCcmEncryptConfig

Parameters included	
Parameter name	**Multiplicity**
CryCcmEncryptSymBlockEncryptCfgRef	1..1
CryCCMMacLength	1..1
CryCCMLengthFieldSize	1..1
CryCCMAdataLen	1..1

Parameter Name	CryCcmEncryptSymBlockEncryptCfgRef
Label	SymBlockEncrypt configuration
Description	Symmetrical block encryption configuration used
Multiplicity	1..1
Type	REFERENCE

Parameter Name	CryCCMMacLength
Label	MAC length
Description	The octet length of the MAC. t parameter in the "NIST Special Publication 800-38C".
Multiplicity	1..1
Type	ENUMERATION
Default value	CRY_CCM_MAC_8_BYTES
Range	CRY_CCM_MAC_4_BYTES CRY_CCM_MAC_6_BYTES CRY_CCM_MAC_8_BYTES CRY_CCM_MAC_10_BYTES CRY_CCM_MAC_12_BYTES CRY_CCM_MAC_14_BYTES CRY_CCM_MAC_16_BYTES

Parameter Name	CryCCMLengthFieldSize
Label	Size of length field
Description	The octet length of the binary representation of the octet length of the payload. q parameter in the "NIST Special Publication 800-38C".
Multiplicity	1..1
Type	ENUMERATION
Default value	CRY_CCM_LEN_FIELD_2_BYTES
Range	CRY_CCM_LEN_FIELD_2_BYTES CRY_CCM_LEN_FIELD_3_BYTES CRY_CCM_LEN_FIELD_4_BYTES

Parameter Name	CryCCMAdataLen

Label	Authenticated data length
Description	The length of the authenticated data
Multiplicity	1..1
Type	INTEGER
Default value	8
Range	>=0 <=4294967296

4.1.45. CryCcmDecrypt

Containers included		
Container name	**Multiplicity**	**Description**
CryCcmDecryptConfig	0..32	

4.1.46. CryCcmDecryptConfig

Parameters included	
Parameter name	**Multiplicity**
CryCcmEncryptSymBlockEncryptCfgRef	1..1
CryCCMMacLength	1..1
CryCCMLengthFieldSize	1..1
CryCCMAdataLen	1..1
CryCCMMaxPayloadSize	1..1

Parameter Name	**CryCcmEncryptSymBlockEncryptCfgRef**
Label	SymBlockEncrypt configuration
Description	Symmetrical block encryption configuration used
Multiplicity	1..1
Type	REFERENCE

Parameter Name	**CryCCMMacLength**
Label	MAC length
Description	The octet length of the MAC.

	t parameter in the "NIST Special Publication 800-38C".
Multiplicity	1..1
Type	ENUMERATION
Default value	CRY_CCM_MAC_8_BYTES
Range	CRY_CCM_MAC_4_BYTES CRY_CCM_MAC_6_BYTES CRY_CCM_MAC_8_BYTES CRY_CCM_MAC_10_BYTES CRY_CCM_MAC_12_BYTES CRY_CCM_MAC_14_BYTES CRY_CCM_MAC_16_BYTES

Parameter Name	**CryCCMLengthFieldSize**
Label	Size of length field
Description	The octet length of the binary representation of the octet length of the payload. q parameter in the "NIST Special Publication 800-38C".
Multiplicity	1..1
Type	ENUMERATION
Default value	CRY_CCM_LEN_FIELD_2_BYTES
Range	CRY_CCM_LEN_FIELD_2_BYTES CRY_CCM_LEN_FIELD_3_BYTES CRY_CCM_LEN_FIELD_4_BYTES

Parameter Name	**CryCCMAdataLen**
Label	Authenticated data length
Description	The length of the authenticated data
Multiplicity	1..1
Type	INTEGER
Default value	8
Range	>=0 <=4294967296

Parameter Name	**CryCCMMaxPayloadSize**

Label	Maximum payload length
Description	The maximum payload data length in bytes. Please note that the memory consumption of the algorithm is proportional with this value.
Multiplicity	1..1
Type	INTEGER
Default value	8
Range	>=0 --- <=4294967296

4.1.47. CryHMACVrfy

Containers included		
Container name	**Multiplicity**	**Description**
CryHMACVrfyConfig	0..32	

Parameters included	
Parameter name	**Multiplicity**
CryHMACVrfyImmediateRestartEnabled	1..1

Parameter Name	**CryHMACVrfyImmediateRestartEnabled**
Label	Enable the cancelation of ongoing requests
Description	Enable the cancelation of an ongoing calculation regardless of the configuration ID.
Multiplicity	1..1
Type	BOOLEAN
Default value	false

4.1.48. CryHMACVrfyConfig

Parameters included	
Parameter name	**Multiplicity**

Parameters included	
CryHMACVrfyHashCfgRef	1..1
CryHMACKeyLength	1..1

Parameter Name	CryHMACVrfyHashCfgRef
Label	Hash configuration
Description	Hash configuration used
Multiplicity	1..1
Type	REFERENCE

Parameter Name	CryHMACKeyLength
Label	Key Length
Description	Key Length used in the HMAC verification
Multiplicity	1..1
Type	INTEGER
Default value	64

4.1.49. CryHMACGen

Containers included		
Container name	Multiplicity	Description
CryHMACGenConfig	0..32	

Parameters included	
Parameter name	Multiplicity
CryHMACGenImmediateRestartEnabled	1..1

Parameter Name	CryHMACGenImmediateRestartEnabled
Label	Enable the cancelation of ongoing requests
Description	Enable the cancelation of an ongoing calculation regardless of the configuration ID.
Multiplicity	1..1
Type	BOOLEAN
Default value	false

4.1.50. CryHMACGenConfig

Parameters included	
Parameter name	**Multiplicity**
CryHMACGenHashCfgRef	1..1
CryHMACKeyLength	1..1

Parameter Name	CryHMACGenHashCfgRef
Label	Hash configuration
Description	Hash configuration used
Multiplicity	1..1
Type	REFERENCE

Parameter Name	CryHMACKeyLength
Label	Key Length
Description	Key Length used in the HMAC generation
Multiplicity	1..1
Type	INTEGER
Default value	64

4.1.51. CryExtractRsaPublicKey

Containers included		
Container name	**Multiplicity**	**Description**
CryExtractRsaPublicKeyConfig	0..32	

4.1.52. CryExtractRsaPublicKeyConfig

Parameters included	
Parameter name	**Multiplicity**
CryExtractRsaPublicKeyLength	1..1

Parameter Name	CryExtractRsaPublicKeyLength

Label	Key Length
Description	Key Length used
Multiplicity	1..1
Type	INTEGER
Default value	384

4.1.53. CryCrc

Containers included		
Container name	**Multiplicity**	**Description**
CryCrcConfig	0..32	

4.1.54. CryCrcConfig

Parameters included	
Parameter name	**Multiplicity**
CryCrcType	1..1
CryCrcPoly	1..1
CryCrcIputXor	1..1
CryCrcOputXor	1..1
CryCrcIputReflect	1..1
CryCrcOputReflect	1..1
CryCrcMaxBytesPerCycle	1..1

Parameter Name	**CryCrcType**
Label	Order of Polynom
Description	Order of polynom that is used for cyclic redundancy check
Multiplicity	1..1
Type	ENUMERATION
Default value	CRY_CRC_32
Range	CRY_CRC_8
	CRY_CRC_16

	CRY_CRC_32

Parameter Name	CryCrcPoly
Label	Generator Polynom
Description	Generator Polynom that is used for cyclic redundancy check
Multiplicity	1..1
Type	INTEGER
Default value	1
Range	>=0 \<=4294967295

Parameter Name	CryCrcIputXor
Label	XOR Input
Description	Polynom with which the input data is XORed
Multiplicity	1..1
Type	INTEGER
Default value	0
Range	>=0 \<=4294967295

Parameter Name	CryCrcOputXor
Label	XOR Output
Description	Polynom with which the output data is XORed
Multiplicity	1..1
Type	INTEGER
Default value	0
Range	>=0 \<=4294967295

Parameter Name	CryCrcIputReflect
Label	Reflect Input
Description	Shall the input data be reflected?
Multiplicity	1..1
Type	BOOLEAN

Default value	false

Parameter Name	**CryCrcOputReflect**
Label	Reflect Output
Description	Shall the output data be reflected?
Multiplicity	1..1
Type	BOOLEAN
Default value	false

Parameter Name	**CryCrcMaxBytesPerCycle**
Label	Bytes per Cycle
Description	maximal number of Bytes given by set of Bytes by Csm_ChecksumUpdate that are processed during a single task cycle
Multiplicity	1..1
Type	INTEGER
Default value	32
Range	>=1 \<=4294967295

4.1.55. CryExtractECPubKey

Containers included		
Container name	**Multiplicity**	**Description**
CryExtractECPubKeyConfig	0..32	

4.1.56. CryExtractECPubKeyConfig

Parameters included	
Parameter name	**Multiplicity**
CryEllipticCurve	1..1

Parameter Name	**CryEllipticCurve**
Label	Elliptic Curve

Description	Elliptic curve used
Multiplicity	1..1
Type	ENUMERATION
Default value	CRY_EDC_255
Range	CRY_EDC_255

4.1.57. PublishedInformation

Parameters included	
Parameter name	**Multiplicity**
PbcfgMSupport	1..1

Parameter Name	**PbcfgMSupport**
Label	PbcfgM support
Description	Specifies whether or not the Cry can use the PbcfgM module for post-build support.
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

4.2. Application programming interface (API)

4.2.1. Type definitions

4.2.1.1. Cry_EdDSAGenKeyType

Purpose	The private key used for creating a signature.
Type	struct

Members	`uint32 length`	Dummy element to make the type compatible with the Csm key type.
	`Cry_LNWordType D`	The secret which has to be a b-bit string.

4.2.1.2. Cry_EdDSAVrfyKeyType

Purpose	The public key used for verifying a signature.	
Type	struct	
Members	`uint32 length`	Dummy element to make the type compatible with the Csm key type.
	`Cry_LNWordType A`	An EdDSA public key is a curve point, which is created by multiplying the s with Point B, encoded in b bits.

4.2.1.3. Cry_ExtractECPubKeyCtxType

Purpose	Structure which contains the context of the elliptic curve public key extraction.	
Type	struct	
Members	`uint32 keyLvl`	Stores the ammount of bytes currently copied in Cry's internal buffer.
	`uint32 keyLen`	Stores the length of the key provided by the user during the initialization of the algorithm.
	`Cry_ECKeyType * oputDataPtr`	Stores the address of the user-provided output buffer.
	`Cry_ExtractECPubKeyStateType ctxState`	Stores the internal state of the elliptic curve public key extraction algorithm.
	`Csm_ReturnType ctxError`	
	`uint8 keyBuf`	

4.2.1.4. Cry_ExtractECPubKeyStateType

Purpose	Internal states of the elliptic curve public key extraction.
Type	enum

Constants	CRY_EXTRACT_ECPUBKEY_STATE_-IDLE	The algorithm is in the idle state.
	CRY_EXTRACT_ECPUBKEY_STATE_-START	The initialization of the algorithm is in progress.
	CRY_EXTRACT_ECPUBKEY_STATE_-INITIALIZED	The initialization of the algorithm is completed.
	CRY_EXTRACT_ECPUBKEY_STATE_-UPDATE	The algorithm was provided with input data.
	CRY_EXTRACT_ECPUBKEY_STATE_-CALCULATED	The algorithm finished copying data to the internal buffer and is waiting for the user-provided output buffer.
	CRY_EXTRACT_ECPUBKEY_STATE_-FINISH	The algorithm is converting the user-provided byte array to long number representation.
	CRY_EXTRACT_ECPUBKEY_STATE_-ERROR	An error occurred due to the user provided data.

4.2.2. Macro constants

4.2.2.1. CRY_MD5_BLOCK_SIZE

Purpose	A message which should be hashed is divided into blocks which are processed individually. This macro gives the length of a block in bytes.
Value	64U

4.2.2.2. CRY_MD5_HASH_LEN_BYTES

Purpose	The length of a message digest computed with the MD5 algorithm in bytes.
Value	16U

4.2.2.3. CRY_MD5_STATE_BYTES

Purpose	The hash algorithm uses a state which is modified by processing the input. The state after processing the whole input is the hash digest. This macro gives the length of the state in 8-bit words.

Value	16U

4.2.2.4. CRY_MD5_STATE_WORDS

Purpose	The hash algorithm uses a state which is modified by processing the input. The state after processing the whole input is the hash digest. This macro gives the length of the state in 32-bit words.
Value	4U

4.2.3. Functions

4.2.3.1. Cry_AES_DecryptFinish

Purpose	Finish AES block decryption computation.	
Synopsis	`Csm_ReturnType` **`Cry_AES_DecryptFinish`** `( void );`	
Return Value	Error code	
	`CSM_E_OK`	If the finish was successfully requested.
	`CSM_E_NOT_OK`	If no AES block decryption computation has been started via Csm_SymBlockDecryptStart(), yet.
	`CSM_E_BUSY`	If the main function is currently processing a requested service.
Description	This function requests the finishing of the AES block decryption computation and the storing of the decrypted text in the given buffer. The finish is performed in Cry_AES_Decrypt_MainFunction().	

4.2.3.2. Cry_AES_DecryptMainFunction

Purpose	Perform the AES block decryption computation tasks.
Synopsis	`void` **`Cry_AES_DecryptMainFunction`** `( void );`
Description	This function performs the actual AES block decryption computation tasks which have been requested by the service functions. The function calls the main function of the configured primitive to perform the tasks.

4.2.3.3. Cry_AES_DecryptStart

Purpose	Start AES block decryption computation.	
Synopsis	`Csm_ReturnType` **`Cry_AES_DecryptStart`** `( const void * cfgPtr , const Csm_SymKeyType * keyPtr );`	
Parameters (in)	`cfgPtr`	A pointer to the configuration for which the start of the AES decryption is requested.
	`keyPtr`	A pointer to the key which should be used in the AES block decryption computation.
Return Value	Error code	
	`CSM_E_OK`	If the start was successfully requested.
	`CSM_E_NOT_OK`	---
	`CSM_E_BUSY`	If a service of the AES block decryption computation is already running.
Description	This function requests the start of the AES decryption computation for the given configuration. The start is performed in Cry_AES_Decrypt_MainFunction().	

4.2.3.4. Cry_AES_DecryptUpdate

Purpose	Update AES block decryption computation.	
Synopsis	`Csm_ReturnType` **`Cry_AES_DecryptUpdate`** `( const uint8 * iputTextPtr , uint32 iputTextLength , uint8 * oputTextPtr , uint32 * oputTextLengthPtr );`	
Parameters (in)	`iputTextPtr`	A pointer to the start of an array which contains the constant cipher text that shall be decrypted.
	`iputTextLength`	Length of the constant cipher text in bytes.
Parameters (in,out)	`oputTextLengthPtr`	Holds a pointer to a memory location in which the length information in bytes is stored. On calling this function this parameter shall contain the size of the buffer provided by oputTextPtr. When the request has finished, the amount of data that has been decrypted shall be stored.
Parameters (out)	`oputTextPtr`	A pointer to the start of an array where the decrypted text will be stored.

Return Value	Error code	
	`CSM_E_OK`	If the update was successfully requested.
	`CSM_E_NOT_OK`	If no AES block decryption computation has been started via Csm_SymBlockDecryptStart(), yet. Or if the provided buffer is too small to store the result.
	`CSM_E_BUSY`	If the main function is currently processing a requested service.
Description	This function requests the update of the AES block decryption computation for the given data. The update is performed in Cry_AES_Decrypt_MainFunction().	

4.2.3.5. Cry_AES_EncryptFinish

Purpose	Finish AES block encryption computation.
Synopsis	`Csm_ReturnType` **`Cry_AES_EncryptFinish`** `( void );`
Return Value	Error code
	`CSM_E_OK` — If the finish was successfully requested.
	`CSM_E_NOT_OK` — If no AES block encryption computation has been started via Csm_SymBlockEncryptStart(), yet.
	`CSM_E_BUSY` — If the main function is currently processing a requested service.
Description	This function requests the finishing of the AES block encryption computation and the storing of the encrypted text in the given buffer. The finish is performed in Cry_AES_Encrypt_MainFunction().

4.2.3.6. Cry_AES_EncryptMainFunction

Purpose	Perform the AES block encryption computation tasks.
Synopsis	`void` **`Cry_AES_EncryptMainFunction`** `( void );`
Description	This function performs the actual AES block encryption computation tasks which have been requested by the service functions. The function calls the main function of the configured primitive to perform the tasks.

4.2.3.7. Cry_AES_EncryptStart

Purpose	Start AES block encryption computation.	
Synopsis	`Csm_ReturnType` **`Cry_AES_EncryptStart`** `( const void * cfgPtr , const Csm_SymKeyType * keyPtr );`	
Parameters (in)	`cfgPtr`	A pointer to the configuration for which the start of the AES encryption is requested.
	`keyPtr`	A pointer to the key which should be used in the AES block encryption computation.
Return Value	Error code	
	`CSM_E_OK`	If the start was successfully requested.
	`CSM_E_NOT_OK`	---
	`CSM_E_BUSY`	If a service of the AES block encryption computation is already running.
Description	This function requests the start of the AES encryption computation for the given configuration. The start is performed in Cry_AES_Encrypt_MainFunction().	

4.2.3.8. Cry_AES_EncryptUpdate

Purpose	Update AES block encryption computation.	
Synopsis	`Csm_ReturnType` **`Cry_AES_EncryptUpdate`** `( const uint8 * iputTextPtr , uint32 iputTextLength , uint8 * oputTextPtr , uint32 * oputTextLengthPtr );`	
Parameters (in)	`iputTextPtr`	A pointer to the start of an array which contains the constant plain text that shall be encrypted.
	`iputTextLength`	Length of the constant plain text in bytes.
Parameters (in,out)	`oputTextLengthPtr`	Holds a pointer to a memory location in which the length information in bytes is stored. On calling this function this parameter shall contain the size of the buffer provided by oputTextPtr. When the request has finished, the amount of data that has been encrypted shall be stored.
Parameters (out)	`oputTextPtr`	A pointer to the start of an array where the encrypted text will be stored.

Return Value	Error code	
	`CSM_E_OK`	If the update was successfully requested.
	`CSM_E_NOT_OK`	If no AES block encryption computation has been started via Csm_SymBlockEncryptStart(), yet. Or if the provided buffer is too small to store the result.
	`CSM_E_BUSY`	If the main function is currently processing a requested service.
Description	This function requests the update of the AES block encryption computation for the given data. The update is performed in Cry_AES_Encrypt_MainFunction().	

4.2.3.9. Cry_CMACGenFinish

Purpose	Finish CMAC generation.	
Synopsis	`Csm_ReturnType` **`Cry_CMACGenFinish`** `( uint8 * resultPtr , uint32 * resultLengthPtr , boolean TruncationIsAllowed );`	
Sync/Async	Asynchronous	
Reentrancy	Not reentrant	
Parameters (in)	`TruncationIsAllowed`	Is truncation of the result allowed or must an error be returned when the result buffer is too small?
Parameters (in,out)	`resultLengthPtr`	A pointer to a variable which contains the maximal allowed length for the CMAC in bits and where the actual length of the CMAC will be stored.
Parameters (out)	`resultPtr`	A pointer to the start of a buffer where the generated CMAC will be stored.
Return Value	Error code	
	`CSM_E_OK`	If the finishing was successfully requested.
	`CSM_E_BUSY`	If the main function is processing a requested service at the moment.
	`CSM_E_NOT_OK`	If no CMAC generation has been started via Cry_CMACGenStart() yet or if the finishing of the CMAC computation is already requested.

Description	This function requests the finishing of the CMAC generation and the storing of the CMAC in the given result buffer. The finishing is performed in Cry_CMACGenMainFunction().

4.2.3.10. Cry_CMACGenMainFunction

Purpose	Perform the CMAC generation tasks.	
Synopsis	`void Cry_CMACGenMainFunction ( void );`	
Sync/Async	Asynchronous	
Reentrancy	Not reentrant	
Description	This function performs the actual CMAC generation tasks which have been requested by the service functions. When a task has finished, the function Csm_MacGenerateServiceCallbackNotification() is called to inform the CSM of the result. If the complete CMAC generation has finished, additionally the function Csm_MacGenerateServiceFinishNotification() is called.	

4.2.3.11. Cry_CMACGenStart

Purpose	Start CMAC generation.	
Synopsis	`Csm_ReturnType Cry_CMACGenStart ( const void * cfgPtr , const Csm_SymKeyType * keyPtr );`	
Sync/Async	Asynchronous	
Reentrancy	Not reentrant	
Parameters (in)	`cfgPtr`	a pointer to the configuration for which the start of the CMAC generation is requested.
	`keyPtr`	a pointer to the key which will be used in the CMAC generation.
Return Value	Error code	
	`CSM_E_OK`	If the start was successfully requested.
	`CSM_E_BUSY`	If a service of the CMAC generation is already running.
Description	This function requests the start of the CMAC generation for the given configuration and key. The start is performed in Cry_CMACGenMainFunction().	

4.2.3.12. Cry_CMACGenUpdate

Purpose	Update CMAC generation.	
Synopsis	`Csm_ReturnType Cry_CMACGenUpdate ( const uint8 * dataPtr , uint32 dataLength );`	
Sync/Async	Asynchronous	
Reentrancy	Not reentrant	
Parameters (in)	`dataPtr`	A pointer to the start of an array which contains a part of the data for which the CMAC will be generated.
	`dataLength`	The amount of data in bytes.
Return Value	Error code	
	`CSM_E_OK`	If the update was successfully requested.
	`CSM_E_BUSY`	If the main function is processing a requested service at the moment.
	`CSM_E_NOT_OK`	If no CMAC generation has been started via Cry_CMACGenStart() yet or if the finishing of the CMAC computation is already requested.
Description	This function requests the update of the CMAC generation for the given data. The update is performed in Cry_CMACGenMainFunction().	

4.2.3.13. Cry_CMACVrfyFinish

Purpose	Finish CMAC verification.	
Synopsis	`Csm_ReturnType Cry_CMACVrfyFinish ( const uint8 * authenticationPtr , uint32 authenticationLength , Csm_VerifyResultType * resultPtr );`	
Sync/Async	Asynchronous	
Reentrancy	Not reentrant	
Parameters (in)	`authenticationPtr`	A pointer to the start of a buffer where the CMAC to verify is stored.
	`authenticationLength`	The length of the CMAC to verify in bits.
Parameters (out)	`resultPtr`	A pointer to a variable where the result of the CMAC verification will be stored.

Return Value	Error code	
	`CSM_E_OK`	If the finishing was successfully requested.
	`CSM_E_BUSY`	If the main function is processing a requested service at the moment.
	`CSM_E_NOT_OK`	If no CMAC verification has been started via Cry_CMACVrfyStart() yet or if the finishing of the CMAC computation is already requested.
Description	This function requests the finishing of the CMAC verification. The finishing is performed in Cry_CMACVrfyMainFunction().	

4.2.3.14. Cry_CMACVrfyMainFunction

Purpose	Perform the CMAC verification tasks.
Synopsis	`void Cry_CMACVrfyMainFunction ( void );`
Sync/Async	Asynchronous
Reentrancy	Not reentrant
Description	This function performs the actual CMAC verification tasks which have been requested by the service functions. When a task has finished, the function Csm_MacVerifyServiceCallbackNotification() is called to inform the CSM of the result. If the complete CMAC verification has finished, additionally the function Csm_MacVerifyServiceFinishNotification() is called.

4.2.3.15. Cry_CMACVrfyStart

Purpose	Start CMAC verification.	
Synopsis	`Csm_ReturnType Cry_CMACVrfyStart ( const void * cfgPtr , const Csm_SymKeyType * keyPtr );`	
Sync/Async	Asynchronous	
Reentrancy	Not reentrant	
Parameters (in)	`cfgPtr`	a pointer to the configuration for which the start of the CMAC verification is requested.

	keyPtr	a pointer to the key which will be used in the CMAC verification.
Return Value	Error code	
	CSM_E_OK	If the start was successfully requested.
	CSM_E_BUSY	If a service of the CMAC verification is already running.
Description	This function requests the start of the CMAC verification for the given configuration and key. The start is performed in Cry_CMACVrfyMainFunction().	

4.2.3.16. Cry_CMACVrfyUpdate

Purpose	Update CMAC verification.	
Synopsis	Csm_ReturnType **Cry_CMACVrfyUpdate** (const uint8 * dataPtr , uint32 dataLength);	
Sync/Async	Asynchronous	
Reentrancy	Not reentrant	
Parameters (in)	dataPtr	A pointer to the start of an array which contains a part of the data for which the CMAC will be verified.
	dataLength	The amount of data in bytes.
Return Value	Error code	
	CSM_E_OK	If the update was successfully requested.
	CSM_E_BUSY	If the main function is processing a requested service at the moment.
	CSM_E_NOT_OK	If no CMAC verification has been started via Cry_CMACVrfyStart() yet or if the finishing of the CMAC computation is already requested.
Description	This function requests the update of the CMAC verification for the given data. The update is performed in Cry_CMACVrfyMainFunction().	

4.2.3.17. Cry_CRCFinish

Purpose	Finish CRC calculation.

Synopsis	`Csm_ReturnType` **`Cry_CRCFinish`** `( uint8 * resultPtr , uint32 * resultLengthPtr , boolean truncationIsAllowed );`	
Parameters (in)	`truncationIsAllowed`	A flag that states wheter a truncation of the calculated CRC result is allowed. TRUE = truncation is allowed. FALSE = truncation is not allowed.
Parameters (out)	`resultPtr`	A pointer to the buffer where the calculated CRC result should be stored. If the result does not fit into the given buffer, and truncation is allowed, the result will be truncated.
	`resultLengthPtr`	On calling the function it is the amount of bytes of resultPtr. After finishing of function call it is the actual length of the calculated CRC result.
Return Value	Error value.	
	`CSM_E_OK`	If the finish was successfully requested.
	`CSM_E_NOT_OK`	If no CRC calculation has been started via Csm_SymKeyExtractStart(), yet.
	`CSM_E_BUSY`	If the main function is currently processing a requested service.
Description	This function requests the finishing of the CRC calculation and the storing of the calculated CRC result in the given buffer. The finish is performed in Cry_CRCMainFunction().	

4.2.3.18. Cry_CRCMainFunction

Purpose	Perform the CRC calculation tasks.
Synopsis	`void` **`Cry_CRCMainFunction`** `( void );`
Description	This function performs the actual CRC calculation tasks which have been requested by the service functions. The function calls the main function of the configured primitive to perform the tasks.

4.2.3.19. Cry_CRCStart

Purpose	Start CRC calculation.

Synopsis	`Csm_ReturnType` **`Cry_CRCStart`** `( const void * cfgPtr );`	
Parameters (in)	`cfgId`	An identification of the configuration for which the start of the CRC calculation should be requested.
Return Value	Error value.	
	`CSM_E_OK`	If the start was successfully requested.
	`CSM_E_NOT_OK`	---
	`CSM_E_BUSY`	If a service of the CRC calculation is already running.
Description	This function requests the start of the CRC calculation for the given configuration. The start is performed in Cry_CRCMainFunction().	

4.2.3.20. Cry_CRCUpdate

Purpose	Update of CRC calculation.	
Synopsis	`Csm_ReturnType` **`Cry_CRCUpdate`** `( const uint8 * dataPtr , uint32 dataLength );`	
Parameters (in)	`dataPtr`	A pointer to the start of an array which contains the data for which the CRC shall be calculated.
	`dataLength`	The amount of bytes of data.
Return Value	Error value.	
	`CSM_E_OK`	If the update was successfully requested.
	`CSM_E_NOT_OK`	If no CRC calculation has been started via Csm_SymKeyExtractStart(), yet.
	`CSM_E_BUSY`	If the main function is currently processing a requested service.
Description	This function requests the update of the CRC calculation for the given data. The update is performed in Cry_CRCMainFunction().	

4.2.3.21. Cry_CVCPublicKeyExtractFinish

Purpose	Finish RSA public key extract computation.

Synopsis	`Csm_ReturnType` **`Cry_CVCPublicKeyExtractFinish`** `( Csm_AsymPublicKeyType * keyPtr );`	
Parameters (out)	`keyPtr`	A pointer to the structure where to store the result.
Return Value	Error code	
	`CSM_E_OK`	If the finish was successfully requested.
	`CSM_E_NOT_OK`	If no RSA public key extract computation was started via Csm_SymKeyExtractStart() beforehand.
	`CSM_E_BUSY`	If the main function is currently processing a requested service.
Description	This function requests the finishing of the RSA public key extract computation and the storing of the extracted key data in the given buffer. The finish is performed in Cry_CVCPublicKeyExtractMainFunction().	

4.2.3.22. Cry_CVCPublicKeyExtractMainFunction

Purpose	Perform the RSA public key extract computation tasks.
Synopsis	`void` **`Cry_CVCPublicKeyExtractMainFunction`** `( void );`
Description	This function performs the actual RSA public key extract computation tasks which have been requested by the service functions.

4.2.3.23. Cry_CVCPublicKeyExtractStart

Purpose	Start RSA public key extract computation.	
Synopsis	`Csm_ReturnType` **`Cry_CVCPublicKeyExtractStart`** `( const void * cfgPtr );`	
Parameters (in)	`cfgPtr`	A pointer to the configuration for which the start of the RSA public key extraction is requested.
Return Value	Error code	
	`CSM_E_OK`	If the start was successfully requested.
	`CSM_E_NOT_OK`	---

	CSM_E_BUSY	If a service of the RSA public key extract computation is already running.
Description	This function requests the start of the RSA public key extract computation for the given configuration. The start is performed in Cry_CVCPublicKeyExtractMainFunction().	

4.2.3.24. Cry_CVCPublicKeyExtractUpdate

Purpose	Update RSA public key extract computation.	
Synopsis	Csm_ReturnType **Cry_CVCPublicKeyExtractUpdate** (const uint8 * dataPtr , uint32 dataLength);	
Parameters (in)	dataPtr	A pointer to the start of an array which contains the key that have to extract in a CSM-conforming format.
	dataLength	Length of the key data in bytes.
Return Value	Error code	
	CSM_E_OK	If the update was successfully requested.
	CSM_E_NOT_OK	If no RSA public key extract computation was started via Csm_SymKeyExtractStart() beforehand.
	CSM_E_BUSY	If the main function is currently processing a requested service.
Description	This function requests the update of the RSA public key extract computation for the given data. The update is performed in Cry_CVCPublicKeyExtractMainFunction(). Update is called twice: 1) Cry_CVCPublicKeyExtractUpdate is first called with the Root certificate (in some documents Verify cv) changing the state of the state machine and preparing Cry_CVCPublicKeyExtractMainFunction to extract the key 2) Another call of the Cry_CVCPublicKeyExtractUpdate is with Signing certificate (in some docs Key cv) changing the state of state machine and preparing Cry_CVCPublicKeyExtractMainFunction for verifying (with the previous extracted key from root cv)	

4.2.3.25. Cry_CbcPkcs7DecryptFinish

Purpose	Finish Decrypt.

Synopsis	`Csm_ReturnType` **`Cry_CbcPkcs7DecryptFinish`** `( uint8 * oputTextPtr , uint32 * oputTextLengthPtr );`	
Parameters (in,out)	`oputTextLengthPtr`	Holds a pointer to a memory location in which the length information in bytes is stored. On calling this function this parameter shall contain the size of the buffer provided by oputTextPtr. When the request has finished, the amount of data that has been decrypted shall be stored.
Parameters (out)	`oputTextPtr`	A pointer to the start of an array where the decrypted text will be stored.
Return Value	Error code	
	`CSM_E_OK`	If the finish was successfully requested.
	`CSM_E_NOT_OK`	If the service has not been started.
	`CSM_E_BUSY`	If another instance of this service is already running.
	`CSM_E_SMALL_BUFFER`	If the result buffer is too small for the encrypted data.
Description	This function finishes the decryption.	

4.2.3.26. Cry_CbcPkcs7DecryptMainFunction

Purpose	Perform primitive tasks.
Synopsis	`void` **`Cry_CbcPkcs7DecryptMainFunction`** `( void );`
Description	This function performs periodic tasks of the primitive that may be necessary (e.g. checking if a CSE command has to be sent or if a CSE command has finished). This function is called by a corresponding CSM main function.

4.2.3.27. Cry_CbcPkcs7DecryptStart

Purpose	Start the symmetric decryption.
Synopsis	`Csm_ReturnType` **`Cry_CbcPkcs7DecryptStart`** `( const void * cfgPtr , const Csm_SymKeyType * keyPtr , const uint8 * initVectorPtr , uint32 initVectorLength );`

Parameters (in)	`cfgPtr`	The service configuration.
	`keyPtr`	Pointer to the key to be used.
	`initVectorPtr`	Holds a pointer to the key which has to be used during the symmetrical decryption computation.
	`initVectorLength`	Holds the length in bytes of the initialisation vector which has to be used during the symmetrical decryption computation.
Return Value	Error code	
	`CSM_E_OK`	If the service can be started.
	`CSM_E_NOT_OK`	If the key is invalid.
	`CSM_E_BUSY`	If another instance of this service is already running.
Description	This function starts the SymDecrypt service which will decrypt blocks of data.	

4.2.3.28. Cry_CbcPkcs7DecryptUpdate

Purpose	Stream data to be decrypted (Single-Shot only).	
Synopsis	`Csm_ReturnType` **`Cry_CbcPkcs7DecryptUpdate`** `( const uint8 * iputTextPtr , uint32 iputTextLength , uint8 * oputTextPtr , uint32 * oputTextLengthPtr );`	
Parameters (in)	`iputTextPtr`	A pointer to the start of an array which contains the constant plain text that shall be decrypted.
	`iputTextLength`	Length of the constant cipher text in bytes.
Parameters (in,out)	`oputTextLengthPtr`	Holds a pointer to a memory location in which the length information in bytes is stored. On calling this function this parameter shall contain the size of the buffer provided by oputTextPtr. When the request has finished, the amount of data that has been decrypted shall be stored.
Parameters (out)	`oputTextPtr`	A pointer to the start of an array where the decrypted text will be stored.
Return Value	Error code	
	`CSM_E_OK`	If the update was successfully requested.

	CSM_E_NOT_OK	If the service has not been started.
	CSM_E_BUSY	If another instance of this service is already running.
	CSM_E_SMALL_BUFFER	If the result buffer is too small for the decrypted data.
Description	This function streams data to be decrypted into the CSE.	

4.2.3.29. Cry_CbcPkcs7EncryptFinish

Purpose	Finish encrypt.	
Synopsis	`Csm_ReturnType Cry_CbcPkcs7EncryptFinish ( uint8 * oputTextPtr , uint32 * oputTextLengthPtr );`	
Parameters (in,out)	oputTextLengthPtr	Holds a pointer to a memory location in which the length information in bytes is stored. On calling this function this parameter shall contain the size of the buffer provided by oputTextPtr. When the request has finished, the amount of data that has been encrypted shall be stored.
Parameters (out)	oputTextPtr	A pointer to the start of an array where the encrypted text will be stored.
Return Value	Error code	
	CSM_E_OK	If the finish was successfully requested.
	CSM_E_NOT_OK	If the service has not been started.
	CSM_E_BUSY	If another instance of this service is already running.
	CSM_E_SMALL_BUFFER	If the result buffer is too small for the encrypted data.
Description	This function finishes the encryption.	

4.2.3.30. Cry_CbcPkcs7EncryptMainFunction

Purpose	Perform primitive tasks.
Synopsis	`void Cry_CbcPkcs7EncryptMainFunction ( void );`

Description	This function performs periodic tasks of the primitive that may be necessary (e.g. checking if a CSE command has to be sent or if a CSE command has finished). This function is called by a corresponding CSM main function.

4.2.3.31. Cry_CbcPkcs7EncryptStart

Purpose	Start the symmetric encryption.	
Synopsis	`Csm_ReturnType` **`Cry_CbcPkcs7EncryptStart`** `( const void * cfgPtr , const Csm_SymKeyType * keyPtr , const uint8 * initVectorPtr , uint32 initVectorLength );`	
Parameters (in)	`cfgPtr`	The service configuration.
	`keyPtr`	Pointer to the key to be used.
	`initVectorPtr`	Holds a pointer to the key which has to be used during the symmetrical encryption computation.
	`initVectorLength`	Holds the length in bytes of the initialisation vector which has to be used during the symmetrical encryption computation.
Return Value	Error code	
	`CSM_E_OK`	If the service can be started.
	`CSM_E_NOT_OK`	If the key is invalid.
	`CSM_E_BUSY`	If another instance of this service is already running.
Description	This function starts the SymEncrypt service which will encrypt blocks of data.	

4.2.3.32. Cry_CbcPkcs7EncryptUpdate

Purpose	Stream data to be encrypted (Single-Shot only).	
Synopsis	`Csm_ReturnType` **`Cry_CbcPkcs7EncryptUpdate`** `( const uint8 * iputTextPtr , uint32 iputTextLength , uint8 * oputTextPtr , uint32 * oputTextLengthPtr );`	
Parameters (in)	`iputTextPtr`	A pointer to the start of an array which contains the constant plain text that shall be encrypted.

	iputTextLength	Length of the constant plain text in bytes.
Parameters (in,out)	oputTextLengthPtr	Holds a pointer to a memory location in which the length information in bytes is stored. On calling this function this parameter shall contain the size of the buffer provided by oputTextPtr. When the request has finished, the amount of data that has been encrypted shall be stored.
Parameters (out)	oputTextPtr	A pointer to the start of an array where the encrypted text will be stored.
Return Value	Error code	
	CSM_E_OK	If the update was successfully requested.
	CSM_E_NOT_OK	If the service has not been started.
	CSM_E_BUSY	If another instance of this service is already running.
	CSM_E_SMALL_BUFFER	If the result buffer is too small for the encrypted data.
Description	This function streams data to be encrypted into the CSE.	

4.2.3.33. Cry_CcmDecryptFinish

Purpose	Finish decrypt.	
Synopsis	Csm_ReturnType **Cry_CcmDecryptFinish** (uint8 * oputTextPtr , uint32 * oputTextLengthPtr);	
Parameters (in,out)	oputTextLengthPtr	Holds a pointer to a memory location in which the length information in bytes is stored. On calling this function this parameter shall contain the size of the buffer provided by oputTextPtr. When the request has finished, the amount of data that has been decrypted shall be stored.
Parameters (out)	oputTextPtr	A pointer to the start of an array where the decrypted text will be stored.
Return Value	Error code	
	CSM_E_OK	If the finish was successfully requested.
	CSM_E_NOT_OK	If the service has not been started.

	CSM_E_BUSY	If another instance of this service is already running.
Description	This function finishes the decryption.	

4.2.3.34. Cry_CcmDecryptMainFunction

Purpose	Perform primitive tasks.	
Synopsis	`void Cry_CcmDecryptMainFunction ( void );`	
Description	This function performs periodic tasks of the primitive that may be necessary (e.g. checking if a CSE command has to be sent or if a CSE command has finished). This function is called by a corresponding CSM main function.	

4.2.3.35. Cry_CcmDecryptStart

Purpose	Start the symmetric decryption.	
Synopsis	`Csm_ReturnType Cry_CcmDecryptStart ( const void * cfgPtr , const Csm_SymKeyType * keyPtr , const uint8 * initVectorPtr , uint32 initVectorLength );`	
Parameters (in)	cfgPtr	The service configuration.
	keyPtr	Pointer to the key to be used.
	initVectorPtr	Holds a pointer to the key which has to be used during the symmetrical decryption computation.
	initVectorLength	Holds the length in bytes of the initialisation vector which has to be used during the symmetrical decryption computation.
Return Value	Error code	
	CSM_E_OK	If the service can be started.
	CSM_E_NOT_OK	If the key is invalid.
Description	This function starts the SymDecrypt service which will decrypt blocks of data.	

4.2.3.36. Cry_CcmDecryptUpdate

Purpose	Stream data to be decrypted (Single-Shot only).

Synopsis	Csm_ReturnType **Cry_CcmDecryptUpdate** (const uint8 * iputTextPtr , uint32 iputTextLength , uint8 * oputTextPtr , uint32 * oputTextLengthPtr);	
Parameters (in)	iputTextPtr	A pointer to the start of an array which contains the constant cipher text that shall be decrypted.
	iputTextLength	Length of the constant cipher text in bytes.
Parameters (in,out)	oputTextLengthPtr	Holds a pointer to a memory location in which the length information in bytes is stored. On calling this function this parameter shall contain the size of the buffer provided by oputTextPtr. When the request has finished, the amount of data that has been decrypted shall be stored.
Parameters (out)	oputTextPtr	A pointer to the start of an array where the decrypted text will be stored.
Return Value	Error code	
	CSM_E_OK	If the update was successfully requested.
	CSM_E_NOT_OK	If the service has not been started.
	CSM_E_BUSY	If another instance of this service is already running.
	CSM_E_SMALL_BUFFER	If the result buffer is too small for the decrypted data.
Description	This function streams data to be decrypted into the CSE.	

4.2.3.37. Cry_CcmEncryptFinish

Purpose	Finish encrypt.	
Synopsis	Csm_ReturnType **Cry_CcmEncryptFinish** (uint8 * oputTextPtr , uint32 * oputTextLengthPtr);	
Parameters (in,out)	oputTextLengthPtr	Holds a pointer to a memory location in which the length information in bytes is stored. On calling this function this parameter shall contain the size of the buffer provided by oputTextPtr. When the request has finished, the amount of data that has been encrypted shall be stored.

Parameters (out)	`oputTextPtr`	A pointer to the start of an array where the encrypted text will be stored.
Return Value	Error code	
	`CSM_E_OK`	If the finish was successfully requested.
	`CSM_E_NOT_OK`	If the service has not been started.
	`CSM_E_BUSY`	If another instance of this service is already running.
Description	This function finishes the encryption.	

4.2.3.38. Cry_CcmEncryptMainFunction

Purpose	Perform primitive tasks.
Synopsis	`void Cry_CcmEncryptMainFunction ( void );`
Description	This function performs periodic tasks of the primitive that may be necessary (e.g. checking if a CSE command has to be sent or if a CSE command has finished). This function is called by a corresponding CSM main function.

4.2.3.39. Cry_CcmEncryptStart

Purpose	Start the symmetric encryption.	
Synopsis	`Csm_ReturnType Cry_CcmEncryptStart ( const void * cfgPtr , const Csm_SymKeyType * keyPtr , const uint8 * initVectorPtr , uint32 initVectorLength );`	
Parameters (in)	`cfgPtr`	The service configuration.
	`keyPtr`	Pointer to the key to be used.
	`initVectorPtr`	Holds a pointer to the key which has to be used during the symmetrical encryption computation.
	`initVectorLength`	Holds the length in bytes of the initialisation vector which has to be used during the symmetrical encryption computation.
Return Value	Error code	
	`CSM_E_OK`	If the service can be started.
	`CSM_E_NOT_OK`	If the key is invalid.

Description	This function starts the SymEncrypt service which will encrypt blocks of data.

4.2.3.40. Cry_CcmEncryptUpdate

Purpose	Stream data to be encrypted (Single-Shot only).	
Synopsis	`Csm_ReturnType Cry_CcmEncryptUpdate ( const uint8 * iputTextPtr , uint32 iputTextLength , uint8 * oputTextPtr , uint32 * oputTextLengthPtr );`	
Parameters (in)	`iputTextPtr`	A pointer to the start of an array which contains the constant associated data and plain text that shall be encrypted.
	`iputTextLength`	Length of the constant plain text in bytes.
Parameters (in,out)	`oputTextLengthPtr`	Holds a pointer to a memory location in which the length information in bytes is stored. On calling this function this parameter shall contain the size of the buffer provided by oputTextPtr. When the request has finished, the amount of data that has been encrypted shall be stored.
Parameters (out)	`oputTextPtr`	A pointer to the start of an array where the encrypted text will be stored.
Return Value	Error code	
	`CSM_E_OK`	If the update was successfully requested.
	`CSM_E_NOT_OK`	If the service has not been started.
	`CSM_E_BUSY`	If another instance of this service is already running.
	`CSM_E_SMALL_BUFFER`	If the result buffer is too small for the encrypted data.
Description	This function streams data to be encrypted into the CSE.	

4.2.3.41. Cry_EdDSAGenFinish

Purpose	Finish signature generation.
Synopsis	`Csm_ReturnType Cry_EdDSAGenFinish ( uint8 * resultPtr , uint32 * resultLengthPtr );`

Parameters (out)	`resultPtr`	A pointer to the start of a buffer which holds the signature.
	`resultLengthPtr`	Length of the signature in bytes.
Return Value	Error value.	
	`CSM_E_OK`	If the finish was successfully requested.
	`CSM_E_NOT_OK`	If no signature generation has been started via Cry_EdDSAGenStart().
	`CSM_E_BUSY`	If the main function is processing a requested service at the moment.
Description	This function requests the finishing of the EdDSA signature generation and the storing of the signature in the given result buffer. The finishing is performed in Cry_EdDSAGenMainFunction().	

4.2.3.42. Cry_EdDSAGenMainFunction

Purpose	Perform the EdDSAGen tasks.
Synopsis	`void Cry_EdDSAGenMainFunction ( void );`
Description	This function performs the actual EdDSA signature generation tasks which have been requested by the service functions. When a task has finished, the function Csm_SignatureGenerateCallbackNotification() is called to inform the CSM of the result. If the complete signature generation has finished, additionally the function Csm_SignatureGenerateServiceFinishNotification() is called.

4.2.3.43. Cry_EdDSAGenStart

Purpose	Start signature generation.	
Synopsis	`Csm_ReturnType Cry_EdDSAGenStart ( const void * cfgPtr , const Csm_AsymPrivateKeyType * privateKeyPtr );`	
Parameters (in)	`cfgPtr`	A pointer to the configuration for which the start of the signature generation should be requested.
	`privateKeyPtr`	A pointer to the key which should be used in the signature generation.
Return Value	Error value.	
	`CSM_E_OK`	If the service can be started.

	CSM_E_BUSY	If another instance of this service is already running.
Description	This function requests the start of the EdDSA signature generation for the given configuration and key. The start is performed in Cry_EdDSAGenMainFunction().	

4.2.3.44. Cry_EdDSAGenUpdate

Purpose	Update signature generation.	
Synopsis	`Csm_ReturnType Cry_EdDSAGenUpdate ( const uint8 * dataPtr , uint32 dataLength );`	
Parameters (in)	dataPtr	A pointer to the start of an array which contains data or a part of the data for which the signature should be created.
	dataLength	The amount of bytes of data.
Return Value	Error value.	
	CSM_E_OK	If the update was successfully requested.
	CSM_E_NOT_OK	If no signature generation has been started via Cry_EdDSAGenStart().
	CSM_E_BUSY	If the main function is processing a requested service at the moment.
Description	This function requests the update of the EdDSA signature generation for the given data. The update is performed in Cry_EdDSAGenMainFunction().	

4.2.3.45. Cry_EdDSAVrfyFinish

Purpose	Finish signature verification.	
Synopsis	`Csm_ReturnType Cry_EdDSAVrfyFinish ( const uint8 * signaturePtr , uint32 signatureLength , Csm_VerifyResultType * resultPtr );`	
Parameters (in)	signaturePtr	A pointer to the start of a buffer which holds the signature.
	signatureLength	Length of the signature in bytes.
Parameters (out)	resultPtr	Verification result.
Return Value	Error value.	

	CSM_E_OK	If the service can be started.
	CSM_E_NOT_OK	If the key is invalid.
	CSM_E_BUSY	If another instance of this service is already running.
Description	This function requests the finishing of the EdDSA signature verification and the storing of the signature in the given result buffer. The finishing is performed in Cry_EdDSAVrfyMainFunction().	

4.2.3.46. Cry_EdDSAVrfyMainFunction

Purpose	Perform the EdDSAVrfy tasks.
Synopsis	`void Cry_EdDSAVrfyMainFunction ( void );`
Description	This function performs the actual EdDSA signature verification tasks which have been requested by the service functions. When a task has finished, the function Csm_SignatureVerifyCallbackNotification() is called to inform the CSM of the result. If the complete signature verification has finished, additionally the function Csm_SignatureVerifyServiceFinishNotification() is called.

4.2.3.47. Cry_EdDSAVrfyStart

Purpose	Start signature verification.	
Synopsis	`Csm_ReturnType Cry_EdDSAVrfyStart ( const void * cfgPtr , const Csm_AsymPublicKeyType * keyPtr );`	
Parameters (in)	cfgPtr	A pointer to the configuration for which the start of the signature verification should be requested.
	keyPtr	A pointer to the key which should be used in the signature verification.
Return Value	Error value.	
	CSM_E_OK	If the service can be started.
	CSM_E_NOT_OK	If the key is invalid.
	CSM_E_BUSY	If another instance of this service is already running.
Description	This function requests the start of the EdDSA signature verification for the given configuration and key. The start is performed in Cry_EdDSAVrfyMainFunction().	

4.2.3.48. Cry_EdDSAVrfyUpdate

Purpose	Update signature verification.	
Synopsis	`Csm_ReturnType` **`Cry_EdDSAVrfyUpdate`** `( const uint8 * dataPtr , uint32 dataLength );`	
Parameters (in)	`dataPtr`	A pointer to the start of an array which contains a part of the data for which the signature should be created.
	`dataLength`	The amount of bytes of data.
Return Value	Error value.	
	`CSM_E_OK`	If the service can be started.
	`CSM_E_NOT_OK`	If the key is invalid.
	`CSM_E_BUSY`	If another instance of this service is already running.
Description	This function requests the update of the EdDSA signature verification for the given data. The update is performed in Cry_EdDSAVrfyMainFunction().	

4.2.3.49. Cry_ExtractECPubKeyFinish

Purpose	Finish asym public key extract computation.	
Synopsis	`Csm_ReturnType` **`Cry_ExtractECPubKeyFinish`** `( Csm_AsymPublicKeyType * keyPtr );`	
Parameters (in)	`keyPtr`	A pointer to the structure where to store the result.
Return Value	Error value.	
	`CSM_E_OK`	If the finish was successfully requested.
	`CSM_E_NOT_OK`	If no asym public key extract computation has been started via Cry_ExtractECPubKeyStart() yet.
	`CSM_E_BUSY`	If the main function is currently processing a requested service.
	`CSM_E_SMALL_BUFFER`	If the size of the structure where to store the result is smaller than the current public key size.

Description	This function requests the finishing of the asym public key extract computation and the storing of the extracted key data in the given buffer. The finish is performed in Cry_ExtractECPubKeyMainFunction().

4.2.3.50. Cry_ExtractECPubKeyMainFunction

Purpose	Perform the asym public key extract computation tasks.
Synopsis	`void Cry_ExtractECPubKeyMainFunction ( void );`
Description	This function performs the actual asym public key extract computation tasks which have been requested by the service functions. The function calls the main function of the configured primitive to perform the tasks.

4.2.3.51. Cry_ExtractECPubKeyStart

Purpose	Start asym public key extract computation.	
Synopsis	`Csm_ReturnType Cry_ExtractECPubKeyStart ( const void * cfgPtr );`	
Parameters (in)	`cfgId`	An identification of the configuration for which the start of the asym public key extract computation should be requested.
Return Value	Error value.	
	`CSM_E_OK`	If the start was successfully requested.
	`CSM_E_NOT_OK`	If the configuration used is not recognised or if the asym public key extract computation has allready been started via Cry_ExtractECPubKeyStart()
	`CSM_E_BUSY`	If a service of the asym public key extract computation is already running.
Description	This function requests the start of the asym public key extract computation for the given configuration. The start is performed in Cry_ExtractECPubKeyMainFunction().	

4.2.3.52. Cry_ExtractECPubKeyUpdate

Purpose	Update asym public key extract computation.

Synopsis	Csm_ReturnType **Cry_ExtractECPubKeyUpdate** (const uint8 * dataPtr , uint32 dataLength);	
Parameters (in)	dataPtr	A pointer to the start of an array which contains the key that have to extract in a CSM-conforming format.
	dataLength	Length of the key in bytes.
Return Value	Error value.	
	CSM_E_OK	If the update was successfully requested.
	CSM_E_NOT_OK	If no asym public key extract computation has been started via Cry_ExtractECPubKeyStart() yet, or if a NULL pointer is assigned as the data array
	CSM_E_BUSY	If the main function is currently processing a requested service.
	CSM_E_SMALL_BUFFER	If the dataLength size is greater then the expected public key size.
Description	This function requests the update of the asym public key extract computation for the given data. The update is performed in Cry_ExtractECPubKeyMainFunction().	

4.2.3.53. Cry_ExtractRsaPublicKeyFinish

Purpose	Finish RSA public key extract computation.	
Synopsis	Csm_ReturnType **Cry_ExtractRsaPublicKeyFinish** (Csm_AsymPublicKeyType * keyPtr);	
Parameters (in)	keyPtr	A pointer to the structure where to store the result.
Return Value	Error value.	
	CSM_E_OK	If the finish was successfully requested.
	CSM_E_NOT_OK	If no RSA public key extract computation has been started via Csm_AsymPublicKeyExtractStart(), yet.
	CSM_E_BUSY	If the main function is currently processing a requested service.
Description	This function requests the finishing of the RSA public key extract computation and the storing of the extracted key data in the given buffer. The finish is performed in Cry_ExtractRsaPublicKeyMainFunction().	

4.2.3.54. Cry_ExtractRsaPublicKeyMainFunction

Purpose	Perform the RSA public key extract computation tasks.
Synopsis	`void` **`Cry_ExtractRsaPublicKeyMainFunction`** `( void );`
Description	This function performs the actual RSA public key extract computation tasks which have been requested by the service functions.

4.2.3.55. Cry_ExtractRsaPublicKeyStart

Purpose	Start RSA public key extract computation.	
Synopsis	`Csm_ReturnType` **`Cry_ExtractRsaPublicKeyStart`** `( const void * cfgPtr );`	
Parameters (in)	`cfgId`	An identification of the configuration for which the start of the RSA public key extract computation should be requested.
Return Value	Error value.	
	`CSM_E_OK`	If the start was successfully requested.
	`CSM_E_NOT_OK`	---
	`CSM_E_BUSY`	If a service of the RSA public key extract computation is already running.
Description	This function requests the start of the RSA public key extract computation for the given configuration. The start is performed in Cry_ExtractRsaPublicKeyMainFunction().	

4.2.3.56. Cry_ExtractRsaPublicKeyUpdate

Purpose	Update RSA public key extract computation.	
Synopsis	`Csm_ReturnType` **`Cry_ExtractRsaPublicKeyUpdate`** `( const uint8 * dataPtr , uint32 dataLength );`	
Parameters (in)	`dataPtr`	A pointer to the start of an array which contains the key that have to extract in a CSM-conforming format.

	`dataLength`	Length of the key in bytes.
Return Value	Error value.	
	`CSM_E_OK`	If the update was successfully requested.
	`CSM_E_NOT_OK`	If no RSA public key extract computation has been started via Csm_AsymPublicKeyExtractStart(), yet.
	`CSM_E_BUSY`	If the main function is currently processing a requested service.
Description	This function requests the update of the RSA public key extract computation for the given data. The update is performed in Cry_ExtractRsaPublicKeyMainFunction().	

4.2.3.57. Cry_HMACGenFinish

Purpose	Finish HMAC generation.	
Synopsis	`Csm_ReturnType Cry_HMACGenFinish ( uint8 * resultPtr , uint32 * resultLengthPtr , boolean TruncationIsAllowed );`	
Sync/Async	Asynchronous	
Reentrancy	Not reentrant	
Parameters (in)	`TruncationIsAllowed`	Is truncation of the result allowed or should an error be returned when the result buffer is too small.
Parameters (in,out)	`resultLengthPtr`	A pointer to a variable which contains the maximal allowed length for the HMAC in bits and where the actual length of the HMAC will be stored.
Parameters (out)	`resultPtr`	A pointer to the start of a buffer where the generated HMAC will be stored.
Return Value	Error code	
	`CSM_E_OK`	If the finish was successfully requested.
	`CSM_E_BUSY`	If the main function is processing a requested service at the moment.
	`CSM_E_NOT_OK`	If no HMAC generation has been started via Cry_HMACGenStart() yet or if the finishing of the HMAC computation is already requested.

Description	This function requests the finishing of the HMAC generation and the storing of the signature in the given result buffer. The finishing is performed in Cry_HMACGenMainFunction().

4.2.3.58. Cry_HMACGenMainFunction

Purpose	Perform the HMAC generation tasks.
Synopsis	`void Cry_HMACGenMainFunction ( void );`
Sync/Async	Asynchronous
Reentrancy	Not reentrant
Description	This function performs the actual HMAC generation tasks which have been requested by the service functions. When a task has finished, the function Csm_MacGenerateServiceCallbackNotification() is called to inform the CSM of the result. If the complete HMAC generation has finished, additionally the function Csm_MacGenerateServiceFinishNotification() is called.

4.2.3.59. Cry_HMACGenStart

Purpose	Start HMAC generation.	
Synopsis	`Csm_ReturnType Cry_HMACGenStart ( const void * cfgPtr , const Csm_SymKeyType * keyPtr );`	
Sync/Async	Asynchronous	
Reentrancy	Not reentrant	
Parameters (in)	`cfgPtr`	A pointer to the configuration for which the start of the signature generation should be requested.
	`keyPtr`	A pointer to the key which should be used in the signature generation.
Return Value	Error code	
	`CSM_E_OK`	If the start was successfully requested.
	`CSM_E_BUSY`	If a service of the HMAC generation is already running.
Description	This function requests the start of the HMAC generation for the given configuration and key. The start is performed in Cry_HMACGenMainFunction().	

4.2.3.60. Cry_HMACGenUpdate

Purpose	Update HMAC generation.	
Synopsis	`Csm_ReturnType Cry_HMACGenUpdate ( const uint8 * dataPtr , uint32 dataLength );`	
Sync/Async	Asynchronous	
Reentrancy	Not reentrant	
Parameters (in)	`dataPtr`	A pointer to the start of an array which contains a part of the data for which the signature should be created.
	`dataLength`	The amount of data in bytes.
Return Value	Error code	
	`CSM_E_OK`	If the update was successfully requested.
	`CSM_E_BUSY`	If the main function is processing a requested service at the moment.
	`CSM_E_NOT_OK`	If no HMAC generation has been started via Cry_HMACGenStart() yet or if the finishing of the HMAC computation is already requested.
Description	This function requests the update of the HMAC generation for the given data. The update is performed in Cry_HMACGenMainFunction().	

4.2.3.61. Cry_HMACVrfyFinish

Purpose	Finish HMAC verification.	
Synopsis	`Csm_ReturnType Cry_HMACVrfyFinish ( const uint8 * authenticationPtr , uint32 authenticationLength , Csm_VerifyResultType * resultPtr );`	
Sync/Async	Asynchronous	
Reentrancy	Not reentrant	
Parameters (in)	`authenticationPtr`	A pointer to the start of a buffer where the generated signature should be stored.

	authenticationLength	A variable which contains the maximal allowed length for the signature and the actual length of the signature.
Parameters (out)	resultPtr	A pointer to a variable where the result of the HMAC verification will be stored.
Return Value	Error code	
	CSM_E_OK	If the finishing was successfully requested.
	CSM_E_BUSY	If the main function is processing a requested service at the moment.
	CSM_E_NOT_OK	If no HMAC verification has been started via Cry_HMACVrfyStart() yet, or if the finishing of the HMAC computation is already requested.
Description	This function requests the finishing of the HMAC verification and the storing of the signature in the given buffer. The finishing is performed in Cry_HMACVrfyMainFunction().	

4.2.3.62. Cry_HMACVrfyMainFunction

Purpose	Perform the HMAC verification tasks.
Synopsis	`void Cry_HMACVrfyMainFunction ( void );`
Sync/Async	Asynchronous
Reentrancy	Not reentrant
Description	This function performs the actual HMAC verification tasks which have been requested by the service functions. When a task has finished, the function Csm_MacVerifyServiceCallbackNotification() is called to inform the CSM of the signature. If the complete signature verification has finished, additionally the function Csm_MacVerifyServiceFinishNotification() is called.

4.2.3.63. Cry_HMACVrfyStart

Purpose	Start HMAC verification.
Synopsis	`Csm_ReturnType Cry_HMACVrfyStart ( const void * cfgPtr , const Csm_SymKeyType * keyPtr );`

Sync/Async	Asynchronous	
Reentrancy	Not reentrant	
Parameters (in)	`cfgPtr`	A pointer to the configuration for which the start of the signature verification is requested.
	`keyPtr`	A pointer to the key which will be used in the HMAC verification.
Return Value	Error code	
	`CSM_E_OK`	If the start was successfully requested.
	`CSM_E_BUSY`	If a service of the HMAC verification is already running.
Description	This function requests the start of the HMAC verification for the given configuration and key. The start is performed in Cry_HMACVrfyMainFunction().	

4.2.3.64. Cry_HMACVrfyUpdate

Purpose	Update HMAC verification.	
Synopsis	`Csm_ReturnType Cry_HMACVrfyUpdate ( const uint8 * dataPtr , uint32 dataLength );`	
Sync/Async	Asynchronous	
Reentrancy	Not reentrant	
Parameters (in)	`dataPtr`	A pointer to the start of an array which contains a part of the data for which the signature should be created.
	`dataLength`	The amount of data in bytes.
Return Value	Error code	
	`CSM_E_OK`	If the update was successfully requested.
	`CSM_E_BUSY`	If the main function is processing a requested service at the moment.
	`CSM_E_NOT_OK`	If no signature verification has been started via Cry_HMACVrfyStart(), yet or if the finishing of the HMAC computation is already requested.
Description	This function requests the update of the HMAC verification for the given data. The update is performed in Cry_HMACVrfyMainFunction().	

4.2.3.65. Cry_KEY_SYM_ExtractFinish

Purpose	Finish KEY_SYM extract computation.	
Synopsis	`Csm_ReturnType` **`Cry_KEY_SYM_ExtractFinish`** `( Csm_SymKeyType * keyPtr );`	
Parameters (out)	`keyPtr`	A pointer to the structure where to store the result.
Return Value	Error code	
	`CSM_E_OK`	If the finish was successfully requested.
	`CSM_E_NOT_OK`	If no KEY_SYM extract computation has been started via Csm_SymKeyExtractStart(), yet.
	`CSM_E_BUSY`	If the main function is currently processing a requested service.
Description	This function requests the finishing of the KEY_SYM extract computation and the storing of the extracted key data in the given buffer. The finish is performed in Cry_KEY_SYM_ExtractMainFunction().	

4.2.3.66. Cry_KEY_SYM_ExtractMainFunction

Purpose	Perform the KEY_SYM extract computation tasks.
Synopsis	`void` **`Cry_KEY_SYM_ExtractMainFunction`** `( void );`
Description	This function performs the actual KEY_SYM extract computation tasks which have been requested by the service functions. The function calls the main function of the configured primitive to perform the tasks.

4.2.3.67. Cry_KEY_SYM_ExtractStart

Purpose	Start KEY_SYM extract computation.
Synopsis	`Csm_ReturnType` **`Cry_KEY_SYM_ExtractStart`** `( const void * cfgPtr );`

Parameters (in)	`cfgPtr`	A pointer to the configuration for which the start of the symmetrical key extraction is requested.
Return Value	Error code	
	`CSM_E_OK`	If the start was successfully requested.
	`CSM_E_NOT_OK`	---
	`CSM_E_BUSY`	If a service of the KEY_SYM extract computation is already running.
Description	This function requests the start of the KEY_SYM extract computation for the given configuration. The start is performed in Cry_KEY_SYM_ExtractMainFunction().	

4.2.3.68. Cry_KEY_SYM_ExtractUpdate

Purpose	Update KEY_SYM extract computation.	
Synopsis	`Csm_ReturnType` **`Cry_KEY_SYM_ExtractUpdate`** `( const uint8 * dataPtr , uint32 dataLength );`	
Parameters (in)	`dataPtr`	A pointer to the start of an array which contains the key that have to extract in a CSM-conforming format.
	`dataLength`	Length of the key in bytes.
Return Value	Error code	
	`CSM_E_OK`	If the update was successfully requested.
	`CSM_E_NOT_OK`	If no KEY_SYM extract computation has been started via Csm_SymKeyExtractStart(), yet.
	`CSM_E_BUSY`	If the main function is currently processing a requested service.
Description	This function requests the update of the KEY_SYM extract computation for the given data. The update is performed in Cry_KEY_SYM_ExtractMainFunction().	

4.2.3.69. Cry_MD5AlgorithmId

Purpose	Algorithm identifier of MD5 hash algorithm.

Synopsis	`const uint8 * Cry_MD5AlgorithmId ( const void * cfgPtr , uint32 * AlgorithmIdLengthPtr );`	
Parameters (in)	`cfgPtr`	a pointer to the configuration for which the algorithm identifier should be returned.
Parameters (out)	`AlgorithmIdLengthPtr`	a pointer to a variable where the actual length of the algorithm identifier should be stored.
Return Value	AlgorithmId A pointer to the start of a byte array which contains the algorithm identifier.	
Description	This function returns the algorithm identifier of the hash algorithm defined in the given configuration.	

4.2.3.70. Cry_MD5Finish

Purpose	Finish function of the MD5 algorithm.	
Synopsis	`Csm_ReturnType Cry_MD5Finish ( uint8 * resultPtr , uint32 * resultLengthPtr , boolean TruncationIsAllowed );`	
Parameters (in)	`inout]`	resultLengthPtr Holds a pointer to the memory location in where the length information is stored. On calling this function this parameter shall contain the size of the buffer provided by resultPtr. When the request has finished, the actual length of the returned value shall be stored.
	`TruncationIsAllowed`	This parameter states whether a truncation of the result is allowed or not. TRUE: truncation is allowed. FALSE: truncation is not allowed.
Parameters (in,out)	`inout]`	resultLengthPtr Holds a pointer to the memory location in where the length information is stored. On calling this function this parameter shall contain the size of the buffer provided by resultPtr. When the request has finished, the actual length of the returned value shall be stored.
Parameters (out)	`resultPtr`	Holds a pointer to the memory location which will hold the result of the hash value computation. If the result does not fit into

		the given buffer, and truncation is allowed, the result shall be truncated.
Return Value	whether the request was successful	
	CSM_E_OK	request successful
	CSM_E_NOT_OK	request failed
	CSM_E_BUSY	the request failed, the MD5 is busy
	CSM_E_SMALL_BUFFER	the provided buffer is too small to store the result, and truncation was not allowed.

4.2.3.71. Cry_MD5MainFunction

Purpose	
Synopsis	`void Cry_MD5MainFunction ( void );`

4.2.3.72. Cry_MD5Start

Purpose	Init function of the MD5 algorithm.	
Synopsis	`Csm_ReturnType Cry_MD5Start ( const void * cfgPtr );`	
Parameters (in)	cfgPtr	Pointer to a Cry MD5 Configuration
Return Value	whether the initialization of the MD5 algorithm was successful	
	CSM_E_OK	the initialization of the MD5 algorithm was successful
	CSM_E_NOT_OK	the initialization of the MD5 algorithm was failed
	CSM_E_BUSY	the initialization failed, the MD5 is busy

4.2.3.73. Cry_MD5Update

Purpose	Update function of the MD5 algorithm.	
Synopsis	`Csm_ReturnType Cry_MD5Update ( const uint8 * dataPtr , uint32 dataLength );`	
Parameters (in)	dataPtr	Holds a pointer to the data to be hashed
	dataLength	Contains the number of bytes to be hashed.

Return Value	whether the request was successful	
	`CSM_E_OK`	request successful
	`CSM_E_NOT_OK`	request failed
	`CSM_E_BUSY`	the request failed, the MD5 is busy

4.2.3.74. Cry_RsaSsaPssVerifyFinish

Purpose	Finish signature verification.	
Synopsis	`Csm_ReturnType` **`Cry_RsaSsaPssVerifyFinish`** `( const uint8 * signaturePtr , uint32 signatureLength , Csm_VerifyResultType * resultPtr );`	
Parameters (in)	`signaturePtr`	a pointer to the start of a buffer which holds the signature.
	`signatureLength`	Length of the signature in bytes.
Parameters (out)	`resultPtr`	Verification result.
Return Value	Error code	
	`CSM_E_OK`	If the finish was successfully requested.
	`CSM_E_BUSY`	If the main function is processing a requested service at the moment.
	`CSM_E_NOT_OK`	If no signature verification has been started via Cry_RsaSsaPssVerifyStart(), yet.
Description	This function requests the finishing of the RSASSA-PSS signature verification and the storing of the signature in the given signature buffer. The finishing is performed in Cry_RsaSsaPssVerifyMainFunction().	

4.2.3.75. Cry_RsaSsaPssVerifyMainFunction

Purpose	Perform the RSASSA-PSS-Verify tasks.
Synopsis	`void` **`Cry_RsaSsaPssVerifyMainFunction`** `( void );`
Description	This function performs the actual RSASSA-PSS signature verification tasks which have been requested by the service functions. When a task has finished, the function Csm_SignatureVerifyServiceCallbackNotification() is called to inform the CSM of the signature. If the complete signature verification has finished, additionally the function Csm_SignatureVerifyServiceFinishNotification() is called.

4.2.3.76. Cry_RsaSsaPssVerifyStart

Purpose	Start signature verification.	
Synopsis	`Csm_ReturnType Cry_RsaSsaPssVerifyStart ( const void * cfgPtr , const Csm_AsymPublicKeyType * keyPtr );`	
Parameters (in)	`cfgPtr`	a pointer to the configuration for which the start of the signature verification should be requested.
	`keyPtr`	a pointer to the key which should be used in the signature verification.
Return Value	Error code	
	`CSM_E_OK`	If the update was successfully requested.
	`CSM_E_BUSY`	If a service of the RSASSA-PSS signature verification is already running.
Description	This function requests the start of the RSASSA-PSS signature verification for the given configuration and key. The start is performed in Cry_RsaSsaPssVerifyMainFunction().	

4.2.3.77. Cry_RsaSsaPssVerifyUpdate

Purpose	Update signature verification.	
Synopsis	`Csm_ReturnType Cry_RsaSsaPssVerifyUpdate ( const uint8 * dataPtr , uint32 dataLength );`	
Parameters (in)	`dataPtr`	a pointer to the start of an array which contains a part of the data for which the signature should be created.
	`dataLength`	The amount of bytes of data.
Return Value	Error code	
	`CSM_E_OK`	If the update was successfully requested.
	`CSM_E_BUSY`	If the main function is processing a requested service at the moment.
	`CSM_E_NOT_OK`	If no signature verification has been started via Cry_SigPKCS1VrfyStart(), yet.
Description	This function requests the update of the RSASSA-PSS signature verification for the given data. The update is performed in Cry_SigPKCS1VrfyMainFunction().	

4.2.3.78. Cry_RsaSsaV15GenFinish

Purpose	
Synopsis	Csm_ReturnType **Cry_RsaSsaV15GenFinish** (uint8 * resultPtr , uint32 * resultLengthPtr);
Return Value	

4.2.3.79. Cry_RsaSsaV15GenMainFunction

Purpose	
Synopsis	void **Cry_RsaSsaV15GenMainFunction** (void);

4.2.3.80. Cry_RsaSsaV15GenStart

Purpose	
Synopsis	Csm_ReturnType **Cry_RsaSsaV15GenStart** (const void * cfgPtr , const Csm_AsymPrivateKeyType * keyPtr);
Return Value	

4.2.3.81. Cry_RsaSsaV15GenUpdate

Purpose	
Synopsis	Csm_ReturnType **Cry_RsaSsaV15GenUpdate** (const uint8 * dataPtr , uint32 dataLength);
Return Value	

4.2.3.82. Cry_RsaSsaV15VerifyFinish

Purpose	Finish signature verification.

Synopsis	`Csm_ReturnType` **`Cry_RsaSsaV15VerifyFinish`** `( const uint8 * signaturePtr , uint32 signatureLength , Csm_VerifyResultType * resultPtr );`	
Parameters (in)	`signaturePtr`	a pointer to the start of a buffer which holds the signature.
	`signatureLength`	Length of the signature in bytes.
Parameters (out)	`resultPtr`	Verification result.
Return Value	Error code	
	`CSM_E_OK`	If the finish was successfully requested.
	`CSM_E_BUSY`	If the main function is processing a requested service at the moment.
	`CSM_E_NOT_OK`	If no signature verification has been started via Cry_SigPKCS1VrfyStart(), yet.
Description	This function requests the finishing of the RSASSA-PKCS1-v1_5 signature verification and the storing of the signature in the given signature buffer. The finishing is performed in Cry_SigPKCS1VrfyMainFunction().	

4.2.3.83. Cry_RsaSsaV15VerifyMainFunction

Purpose	Perform the RSASSA-PKCS1-v1_5 verification tasks.
Synopsis	`void` **`Cry_RsaSsaV15VerifyMainFunction`** `( void );`
Description	This function performs the actual RSASSA-PKCS1-v1_5 signature verification tasks which have been requested by the service functions. When a task has finished, the function Csm_SignatureVerifyServiceCallbackNotification() is called to inform the CSM of the signature. If the complete signature verification has finished, additionally the function Csm_SignatureVerifyServiceFinishNotification() is called.

4.2.3.84. Cry_RsaSsaV15VerifyStart

Purpose	Start signature verification.	
Synopsis	`Csm_ReturnType` **`Cry_RsaSsaV15VerifyStart`** `( const void * cfgPtr , const Csm_AsymPublicKeyType * keyPtr );`	
Parameters (in)	`cfgPtr`	a pointer to the configuration for which the start of the signature verification should be requested.

	keyPtr	a pointer to the key which should be used in the signature verification.
Return Value	Error code	
	CSM_E_OK	If the update was successfully requested.
	CSM_E_BUSY	If a service of the RSASSA-PKCS1-v1_5 signature verification is already running.
Description	This function requests the start of the RSASSA-PKCS1-v1_5 signature verification for the given configuration and key. The start is performed in Cry_RsaSsaV15VerifyMainFunction().	

4.2.3.85. Cry_RsaSsaV15VerifyUpdate

Purpose	Update signature verification.	
Synopsis	`Csm_ReturnType Cry_RsaSsaV15VerifyUpdate ( const uint8 * dataPtr , uint32 dataLength );`	
Parameters (in)	dataPtr	a pointer to the start of an array which contains a part of the data for which the signature should be created.
	dataLength	The amount of bytes of data.
Return Value	Error code	
	CSM_E_OK	If the update was successfully requested.
	CSM_E_BUSY	If the main function is processing a requested service at the moment.
	CSM_E_NOT_OK	If no signature verification has been started via Cry_SigPKCS1VrfyStart(), yet.
Description	This function requests the update of the RSASSA-PKCS1-v1_5 signature verification for the given data. The update is performed in Cry_SigPKCS1VrfyMainFunction().	

4.2.3.86. Cry_SHA1AlgorithmId

Purpose	Algorithm identifier of hash algorithm.
Synopsis	`const uint8 * Cry_SHA1AlgorithmId ( const void * cfgPtr , uint32 * AlgorithmIdLengthPtr );`

Parameters (in)	`cfgPtr`	an pointer to the configuration for which the algorithm identifier should be returned
Parameters (out)	`AlgorithmIdLengthPtr`	a pointer to a variable where the actual length of the algorithm identifier should be stored
Return Value	Algorithm Id	
Description	This function returns the algorithm identifier of the hash algorithm defined in the given configuration. See RFC 3279 for examples	

4.2.3.87. Cry_SHA1Finish

Purpose	Finish SHA1 computation.	
Synopsis	`Csm_ReturnType` **`Cry_SHA1Finish`** `( uint8 * resultPtr , uint32 * resultLengthPtr , boolean truncationAllowed );`	
Parameters (in)	`truncationAllowed`	If this flag is TRUE and the hash digest is longer than the given result buffer, the hash is truncated to the buffer length. If this flag is FALSE and the hash digest is longer than the given result buffer, an error code of CSM_E_SMALL_BUFFER is returned
Parameters (in,out)	`resultLengthPtr`	a pointer to a variable which contains the maximal allowed length in bytes for the hash and where the actual length in bytes of the hash should be stored
Parameters (out)	`resultPtr`	Pointer to the start of a buffer where the hash digest should be stored
Return Value	Error code	
	`CSM_E_OK`	If the finish was successfully requested.
	`CSM_E_BUSY`	If the main function is processing a requested service at the moment
	`CSM_E_NOT_OK`	If no SHA computation has been started via Cry_SHA1Start(), yet.
Description	This function requests the finishing of the SHA computation and the storing of the hash digest in the given result buffer. The finishing is performed in Cry_SHA1MainFunction().	

4.2.3.88. Cry_SHA1MainFunction

Purpose	Perform the SHA1 tasks.
Synopsis	`void Cry_SHA1MainFunction ( void );`
Description	This function performs the actual SHA computation tasks which have been requested by the service functions. When a task has finished, the function Csm_HashServiceCallbackNotification() is called to inform the CSM of the result. If the complete SHA1 computation has finished, additionally the function Csm_HashServiceFinishNotification() is called.

4.2.3.89. Cry_SHA1Start

Purpose	Start SHA1 computation.	
Synopsis	`Csm_ReturnType Cry_SHA1Start ( const void * cfgPtr );`	
Parameters (in)	`cfgPtr`	A pointer to the configuration for which the start of the SHA computation should be requested.
Return Value	Error code	
	`CSM_E_OK`	If the start was successfully requested.
	`CSM_E_BUSY`	If a service of the SHA compuation is already running
Description	This function requests the start of the SHA computation for the given configuration. The start is performed in Cry_SHA1MainFunction()	

4.2.3.90. Cry_SHA1Update

Purpose	Update SHA1 computation.	
Synopsis	`Csm_ReturnType Cry_SHA1Update ( const uint8 * dataPtr , uint32 dataLength );`	
Parameters (in)	`dataPtr`	A pointer to the start of an array which contains a part of the data for which the SHA digest should be created.
	`dataLength`	The amount of bytes of data
Return Value	Error code	
	`CSM_E_OK`	If the update was successfully requested.

	CSM_E_BUSY	If the main function is processing a requested service at the moment
	CSM_E_NOT_OK	If no SHA computation has been started via Cry_SHA1Start(), yet.
Description	This function requests the update of the SHA computation for the given data. The update is performed in Cry_SHA1MainFunction().	

4.2.3.91. Cry_SHA2AlgorithmId

Purpose	Algorithm identifier of hash algorithm.	
Synopsis	`const uint8 * Cry_SHA2AlgorithmId ( const void * cfgPtr , uint32 * AlgorithmIdLengthPtr );`	
Parameters (in)	cfgPtr	an pointer to the configuration for which the algorithm identifier should be returned
Parameters (out)	AlgorithmIdLengthPtr	a pointer to a variable where the actual length of the algorithm identifier should be stored
Return Value	Algorithm Id	
Description	This function returns the algorithm identifier of the hash algorithm defined in the given configuration. See RFC 3279 for examples	

4.2.3.92. Cry_SHA2Finish

Purpose	Finish SHA2 computation.	
Synopsis	`Csm_ReturnType Cry_SHA2Finish ( uint8 * resultPtr , uint32 * resultLengthPtr , boolean truncationAllowed );`	
Parameters (in)	truncationAllowed	If this flag is TRUE and the hash digest is longer than the given result buffer, the hash is truncated to the buffer length. If this flag is FALSE and the hash digest is longer than the given result buffer, an error code of CSM_E_SMALL_BUFFER is returned
Parameters (in,out)	resultLengthPtr	a pointer to a variable which contains the maximal allowed length in bytes for the

		hash and where the actual length in bytes of the hash should be stored
Parameters (out)	`resultPtr`	Pointer to the start of a buffer where the hash digest should be stored
Return Value	Error code	
	`CSM_E_OK`	If the finish was successfully requested.
	`CSM_E_BUSY`	If the main function is processing a requested service at the moment
	`CSM_E_NOT_OK`	If no SHA computation has been started via Cry_SHA2Start(), yet.
Description	This function requests the finishing of the SHA computation and the storing of the hash digest in the given result buffer. The finishing is performed in Cry_SHA2MainFunction().	

4.2.3.93. Cry_SHA2MainFunction

Purpose	Perform the SHA2 tasks.
Synopsis	`void Cry_SHA2MainFunction ( void );`
Description	This function performs the actual SHA computation tasks which have been requested by the service functions. When a task has finished, the function Csm_HashServiceCallbackNotification() is called to inform the CSM of the result. If the complete SHA2 computation has finished, additionally the function Csm_HashServiceFinishNotification() is called.

4.2.3.94. Cry_SHA2Start

Purpose	Start SHA2 computation.	
Synopsis	`Csm_ReturnType Cry_SHA2Start ( const void * cfgPtr );`	
Parameters (in)	`cfgPtr`	A pointer to the configuration for which the start of the SHA computation should be requested.
Return Value	Error code	
	`CSM_E_OK`	If the start was successfully requested.
	`CSM_E_BUSY`	If a service of the SHA compuation is already running

Description	This function requests the start of the SHA computation for the given configuration. The start is performed in Cry_SHA2MainFunction()

4.2.3.95. Cry_SHA2Update

Purpose	Update SHA2 computation.	
Synopsis	`Csm_ReturnType Cry_SHA2Update ( const uint8 * dataPtr , uint32 dataLength );`	
Parameters (in)	`dataPtr`	A pointer to the start of an array which contains a part of the data for which the SHA digest should be created.
	`dataLength`	The amount of bytes of data
Return Value	Error code	
	`CSM_E_OK`	If the update was successfully requested.
	`CSM_E_BUSY`	If the main function is processing a requested service at the moment
	`CSM_E_NOT_OK`	If no SHA computation has been started via Cry_SHA2Start(), yet.
Description	This function requests the update of the SHA computation for the given data. The update is performed in Cry_SHA2MainFunction().	

4.2.3.96. Cry_SHA3AlgorithmId

Purpose	Algorithm identifier of hash algorithm.	
Synopsis	`const uint8 * Cry_SHA3AlgorithmId ( const void * ConfigPtr , uint32 * AlgorithmIdLengthPtr );`	
Parameters (in)	`ConfigPtr`	an pointer to the configuration for which the algorithm identifier should be returned
Parameters (out)	`AlgorithmIdLengthPtr`	a pointer to a variable where the actual length of the algorithm identifier should be stored
Return Value	Algorithm Id	
Description	This function returns the algorithm identifier of the hash algorithm defined in the given configuration. See RFC 3279 for examples	

4.2.3.97. Cry_SHA3Finish

Purpose	Finish SHA3 computation.	
Synopsis	`Csm_ReturnType` **`Cry_SHA3Finish`** `( uint8 * resultPtr , uint32 * resultLengthPtr , boolean truncationAllowed );`	
Parameters (in)	`truncationAllowed`	If this flag is TRUE and the hash digest is longer than the given result buffer, the hash is truncated to the buffer length. If this flag is FALSE and the hash digest is longer than the given result buffer, an error code of CSM_E_SMALL_BUFFER is returned
Parameters (in,out)	`resultLengthPtr`	a pointer to a variable which contains the maximal allowed length in bytes for the hash and where the actual length in bytes of the hash should be stored
Parameters (out)	`resultPtr`	Pointer to the start of a buffer where the hash digest should be stored
Return Value	Error code	
	`CSM_E_OK`	If the finish was successfully requested.
	`CSM_E_BUSY`	If the main function is processing a requested service at the moment
	`CSM_E_NOT_OK`	If no SHA computation has been started via Cry_SHA3Start(), yet.
Description	This function requests the finishing of the SHA computation and the storing of the hash digest in the given result buffer. The finishing is performed in Cry_SHA3MainFunction().	

4.2.3.98. Cry_SHA3MainFunction

Purpose	Perform the SHA3 tasks.
Synopsis	`void` **`Cry_SHA3MainFunction`** `( void );`
Description	This function performs the actual SHA computation tasks which have been requested by the service functions. When a task has finished, the function Csm_HashServiceCallbackNotification() is called to inform the CSM of the result. If the complete SHA3

	computation has finished, additionally the function Csm_HashServiceFinishNotification() is called.

4.2.3.99. Cry_SHA3Start

Purpose	Start SHA3 computation.	
Synopsis	`Csm_ReturnType Cry_SHA3Start ( const void * ConfigPtr );`	
Parameters (in)	`ConfigPtr`	A pointer to the configuration for which the start of the SHA computation should be requested.
Return Value	Error code	
	`CSM_E_OK`	If the start was successfully requested.
	`CSM_E_BUSY`	If a service of the SHA compuation is already running
Description	This function requests the start of the SHA computation for the given configuration. The start is performed in Cry_SHA3MainFunction()	

4.2.3.100. Cry_SHA3Update

Purpose	Update SHA3 computation.	
Synopsis	`Csm_ReturnType Cry_SHA3Update ( const uint8 * dataPtr , uint32 dataLength );`	
Parameters (in)	`dataPtr`	A pointer to the start of an array which contains a part of the data for which the SHA digest should be created.
	`dataLength`	The amount of bytes of data
Return Value	Error code	
	`CSM_E_OK`	If the update was successfully requested.
	`CSM_E_BUSY`	If the main function is processing a requested service at the moment
	`CSM_E_NOT_OK`	On invalid input (dataLength != 0 && NULL_PTR == dataLength) or if the SHA3 computation is in the idle state
Description	This function requests the update of the SHA computation for the given data. The update is performed in Cry_SHA3MainFunction().	

4.2.3.101. Cry_SSGGenerate

Purpose	Generate pseudo random bytes.	
Synopsis	`Csm_ReturnType` **`Cry_SSGGenerate`** `( const void * cfgPtr , uint8 * resultPtr , uint32 resultLength );`	
Parameters (in)	`cfgPtr`	A pointer to the configuration which should be used in the pseudo random byte generation. This configuration contains, among others, a pointer to the state which should be used for the generation.
Parameters (out)	`resultPtr`	A pointer to the start of a buffer where the generated pseudo random bytes should be stored
	`resultLength`	Holds the amount of bytes which should be generated
Return Value	Generation operation state	
	`CSM_E_OK`	If the byte generation was successfully requested
	`CSM_E_BUSY`	If another service of the SSG is already running
Description	This function requests the generation of pseudo random bytes. The byte generation is performed in Cry_SSGMainFunction()	

4.2.3.102. Cry_SSGGenerateMainFunction

Purpose	This function performs the actual SSG tasks which have been requested by the service functions. When a task has finished, the function Csm_RandomCallbackNotification() is called to inform the CSM of the result.
Synopsis	`void` **`Cry_SSGGenerateMainFunction`** `( void );`

4.2.3.103. Cry_SSGSeedFinish

Purpose	This function requests the finishing of the seeding of the SSG. The finishing of the seeding is performed in Cry_SSGMainFunction().

Synopsis	`Csm_ReturnType` **`Cry_SSGSeedFinish`** `( void );`	
Return Value	Request result	
	`CSM_E_OK`	If the finish was successfully requested.
	`CSM_E_NOT_OK`	If there was an error
	`CSM_E_BUSY`	If the SSGSeed Service computation is already running.

4.2.3.104. Cry_SSGSeedMainFunction

Purpose	This function performs the actual SSG tasks which have been requested by the service functions. When a task has finished, the function Csm_RandomCallbackNotification() is called to inform the CSM of the result.
Synopsis	`void` **`Cry_SSGSeedMainFunction`** `( void );`

4.2.3.105. Cry_SSGSeedStart

Purpose	This function requests the initialization of the SSG state which has been given with the passed configuration pointer. The initialization is performed in Cry_SSGMainFunction().	
Synopsis	`Csm_ReturnType` **`Cry_SSGSeedStart`** `( const void * cfgPtr );`	
Parameters (in)	`cfgPtr`	Pointer to the configuration which should be used in the state initialization. This configuration contains, among others, a pointer to the state which should be initialized.
Return Value	Request result	
	`CSM_E_OK`	If the start was successfully requested.
	`CSM_E_BUSY`	If a service of SSGSeed is already running.

4.2.3.106. Cry_SSGSeedUpdate

Purpose	This function requests the seeding of the SSG state which has been given with the passed configuration pointer. The seeding is performed in Cry_SSGMainFunction().

Synopsis	`Csm_ReturnType` **`Cry_SSGSeedUpdate`** `( const uint8 * seedPtr , uint32 seedLength );`	
Parameters (in)	`seedPtr`	Pointer to the start of an array which contains the seed.
	`seedLength`	The length of the seed in byte
Return Value	Rquest result	
	`CSM_E_OK`	If the update was successfully requested.
	`CSM_E_NOT_OK`	If there was an error
	`CSM_E_BUSY`	If the SSGSeed Service computation is already running.

4.3. Integration notes

4.3.1. Integration requirements

WARNING 

Integration requirements list is not exhaustive

The following list of integration requirements helps you to integrate your product. However, this list is not exhaustive. You also require information from the user guide, release notes, and EB tresos AutoCore known issues to successfully integrate your product.

Integration requirements are not listed for the Cry module.