

	Document No.	AUTOSAR HaeModule Crypto Driver User Manual	Revision Date	'24. 04
	-		Revision No.	1.03b

AUTOSAR HaeModule Crypto Driver User Manual

HSM and CryptoLib Module



	Document No.	AUTOSAR HaeModule Crypto Driver User Manual	Revision Date	'24. 04
	-		Revision No.	1.03b

VERSION HISTORY

Version	On	What's changing?	Reviewers	Remarks
1.00	2023.11.17	Writing a user manual for the first time	-	
1.01	2023.12.18	Crypto_76_HaeModule v1.0.1 Update	-	
1.02	2023.01.19	Crypto_76_HaeModule v1.0.2 Update	-	
1.02a	2024.03.05	Manual error correction (RSAES-OAEP input struct, curve ID)	-	
1.02b	2024.03.06	Added caveats for using the HSM API	-	
1.02c	2024.03.07	Manual error correction (ED448 input struct, curve ID)	-	
1.02d	2024.03.19	Signature value +1 byte size accepted compose	-	
1.03	2024.03.27	Added instructions for using PBKDF2	JeonJM	
1.03a	2024.04.16	Manual error correction (block cipher key length unit)	-	
1.03b	2024.04.24	KeyId caution added, HSM user caution added	JeonJM	

	Document No.	AUTOSAR HaeModule Crypto Driver User Manual	Revision Date	'24. 04
	-		Revision No.	1.03b

Notes for HSM Users

- For HSM users, WDT timeout can occur depending on the API algorithm for low-performance MCUs , so be sure to increase the task timeout time sufficiently. At this time, the setting time must be set by the user according to the time of the API execution, and the result may vary depending on the controller environment, MCU environment, and user design, so it is necessary to check this directly in the user environment.
This is [detailed in "3.2 Notes"](#).
- CRYPTO_E_BUSY can be returned according to the Autosar standard.**IN THIS CASE, IT IS NOT AN ERROR SITUATION, IT MEANS THAT THE HSM IS ALREADY PERFORMING ANOTHER TASK.** Therefore, it is necessary to allow the API to be called again after a while. At this time, if the API is continuously retried using while, etc., the MCU with low performance may experience the same WDT timeout as above, so it is necessary to design the sequence so that after the end of the TASK, it is possible to enter the TASK again and call the API.
- For the above reasons , it is recommended that the HSM API be performed with Async - SingleCall.**
- All HSM-specific APIs meet the Autosar standard specification and include HSM-specific functional specifications.

	Document No.	AUTOSAR HaeModule Crypto Driver User Manual	Revision Date	'24. 04
	-		Revision No.	1.03b

TURN

NOTES FOR HSM USERS.....3

1. DOCUMENT OVERVIEW 12

1.1. Purpose of the document 12
1.2. Document Scope..... 12
1.3. target..... 12
1.4. Definitions..... 12
1.5. References..... 12

2. HAEMODULE CRYPTO DRIVER OVERVIEW..... 13

2.1. HaeModule Crypto Driver Architecture 13
2.2. HaeModule Crypto Driver Structure..... 14
2.3. Files & Tools Provided 15

2.3.1. Static HaeModule Crypto Driver source file and Header files 15
2.3.2. AUTOSAR_Crypto_76_HaeModule_ECU_Configuration_PDF.arxml 15
2.3.3. Ecud_Crypto_76_HaeModule.arxml 15
2.3.4. Bswmd_Crypto_76_HaeModule.template 15
2.3.5. Crypto Generator Tools 15

3. GENERAL SPECIFICATIONS 16

3.1. Operating Conditions 16
3.2. Notice 16

	Document No.	AUTOSAR HaeModule Crypto Driver User Manual	Revision Date	'24. 04
	-		Revision No.	1.03b

3.3.

Interface Support

17

3.4.

What applies to AUTOSAR Classic Release R19–11

19

4.

CRYPTO GENERATOR TOOLS

20

4.1

Usage and input parameters

20

4.2

Generate Source and Header Files

20

4.3

Creating a BSWMD File

20

5.

CRYPTO DRIVER OBJECTS

21

5.1.

Support Modules

21

5.2.

Module-specific Objects and Primitives

21

5.3.

Module-specific prefix

22

6.

JOB PROCESSING

23

6.1.

Job Primitive Settings by Crypto Algorithm

23

6.1.1.

CRYPTO_HASH

23

6.1.2.

CRYPTO_MACGENERATE

24

6.1.2.1.

TDES CMAC

24

6.1.2.2.

AES CMAC

25

6.1.2.3.

AES GMAC

26

6.1.2.4.

HMAC

28

6.1.3.

CRYPTO_MACVERIFY

29

6.1.3.1.

TDES CMAC

29

6.1.3.2.

AES CMAC

30


6.1.3.3.

AES GMAC

31

	Document No.	AUTOSAR HaeModule Crypto Driver User Manual	Revision Date	'24. 04
	-		Revision No.	1.03b

6.1.3.4.	HMAC	32
6.1.4.	CRYPTO_ENCRYPT	33
6.1.4.1.	TDES.....	33
6.1.4.2.	AES	34
6.1.4.3.	CHACHA20.....	35
6.1.4.4.	RSAES-PKCS1_v1_5.....	36
6.1.4.5.	RSAES-OAEP.....	37
6.1.5.	CRYPTO_DECRYPT	39
6.1.5.1.	TDES.....	39
6.1.5.2.	AES	40
6.1.5.3.	CHACHA20.....	41
6.1.5.4.	RSAES-PKCS1_v1_5.....	42
6.1.5.5.	RSAES-OAEP.....	43
6.1.6.	CRYPTO_AEADENCRYPT	44
6.1.6.1.	AES GCM	44
6.1.6.2.	CHACHA20-POLY1305.....	46
6.1.7.	CRYPTO_AEADDECRYPT.....	47
6.1.7.1.	AES GCM	47
6.1.7.2.	CHACHA20-POLY1305.....	49
6.1.8.	CRYPTO_SIGNATUREGENERATE	50
6.1.8.1.	RSASSA-PKCS1_v1_5.....	50

	Document No.	AUTOSAR HaeModule Crypto Driver User Manual	Revision Date	'24. 04
	-		Revision No.	1.03b

6.1.8.2.	RSASSA-PSS	51
6.1.8.3.	ECDSA	52
6.1.8.4.	EDDSA ED448.....	54
6.1.9.	CRYPTO_SIGNATUREVERIFY	55
6.1.9.1.	RSASSA-PKCS1_v1_5.....	55
6.1.9.2.	RSASSA-PSS	56
6.1.9.3.	ECDSA	57
6.1.9.4.	EDDSA ED448.....	59
6.1.10.	CRYPTO_RANDOMGENERATE	61
6.1.10.1.	DRBG	61
6.1.10.2.	PRNG	61
6.1.10.3.	TRNG	62
6.1.11.	CRYPTO_RANDOMSEED	63
6.1.11.1.	DRBG	63
6.1.12.	CRYPTO_KEYEXCHANGEALCPUBVAL.....	64
6.1.12.1.	Diffie-Hellman	64
6.1.12.2.	ECDH	65
6.1.13.	CRYPTO_KEYEXCHANGEALCSECRET.....	67
6.1.13.1.	Diffie-Hellman	67
6.1.13.2.	ECDH	68
6.1.14.	CRYPTO_KEYDERIVE.....	70

	Document No.	AUTOSAR HaeModule Crypto Driver User Manual	Revision Date	'24. 04
	-		Revision No.	1.03b

6.1.14.1.	PBKDF2	70
6.1.15.	CRYPTO_KEYSETVALID	72
6.2.	How to Add New Users to Crypto Primitive	73
6.3.	Setting up the Crypto MainFunction for the Asynchronous Job.....	74
7.	KEY MANAGEMENT	75
7.1.	Crypto Key Element Settings	75
7.1.1.	Setting up the HSM Crypto Key Element	75
7.1.2.	HSM Crypto Key Element characteristic.....	77
7.2.	How to set CryptoKeyElementValue	78
7.2.1.	How to enter with CryptoKeyElementInitValue	78
7.2.2.	How to enter with the Crypto_KeyElementSet API	78
7.3.	How to Add a New CryptoKey	78
8.	CRYPTO CONFIGURATION	81
8.1.	CryptoGeneral.....	82
8.2.	CryptoDriverObjects.....	83
8.3.	CryptoPrimitives	84
8.4.	CryptoKeys	85
8.5.	CryptoKeyTypes	86
8.6.	CryptoKeyElements	87
9.	CRYPTO DRIVER PORTING GUIDE.....	89
9.1.	mobilgene Classic Platform	89
9.2.	AUTOSAR Platform	89
9.3.	Memory Section	90

	Document No.	AUTOSAR HaeModule Crypto Driver User Manual	Revision Date	'24. 04
	-		Revision No.	1.03b

9.3.1.

Link Script section

90

9.3.2.

Platform memory map file.....

91

APPENDIX. TECHNICAL SUPPORT GUIDE

93

	Document No.	AUTOSAR HaeModule Crypto Driver User Manual	Revision Date	'24. 04
	-		Revision No.	1.03b

PICTURE

Figure 1. AUTOSAR Layered View with HaeModule Crypto Driver 13

Figure 2. Create HaeModule Crypto BSWMD 14

Figure 3. HSM Generated Configuration Files 14

Figure 4. HaeModule Crypto Driver Source Structure 14

Figure 5. HaeModule Crypto Driver Supported Modules 21

Figure 6. Module-specific Objects and Primitives 21

Figure 7. New CryptoPrimitive added 73

Figure 8. HaeModule CryptoKeyElement Container 76

Figure 9. Setting up the HSM Crypto Key Element..... 76

Figure 10. AUTOSAR Crypto Container Structure 81

Figure 11. AUTOSAR CryptoGeneral Container 82

Figure 12. AUTOSAR CryptoDriverObject Container 83

Figure 13. AUTOSAR CryptoPrimitive Container..... 84

Figure 14. AUTOSAR CryptoKey Container 85

Figure 15. AUTOSAR CryptoKeyType Container..... 86

Figure 16. AUTOSAR CryptoKeyElement Container 88

	Document No.	AUTOSAR HaeModule Crypto Driver User Manual	Revision Date	'24. 04
	-		Revision No.	1.03b

TICKET

Table 1. Crypto Driver Interface Support

18

Table 2. Module-specific Objects and Primitives

21

Table 3. Module-specific prefix

22

Table 4. Object-specific Crypto MainFunction.....

74

Table 5. HSM Key Table

75

Table 6. HaeModule CryptoGeneral Container Contents

82

Table 7. HaeModule CryptoDriverObject Container Contents

83

	Document No.	AUTOSAR HaeModule Crypto Driver User Manual	Revision Date	'24. 04
	-		Revision No.	1.03b

1. Document overview

1.1. Purpose of the document

The purpose of this document is to provide a guide for using the AUTOSAR Crypto Driver for our HSMs and CryptoLib modules.

1.2. Document Scope

This document describes the driver's overview, structure, features, and API for using the AUTOSAR Crypto Driver for our HSMs and CryptoLib modules.

1.3. target

This document is intended for developers who are responsible for developing ECUs that perform security functions with their HSM modules or CryptoLib.Users of this document should be familiar with the manual for the Autosar standard specification in advance.

1.4. Definitions

Going forward, we will use the term HaeModule when referring to both our HSM and CryptoLib modules, and we will use the term HSM module or CryptoLib module when referring to individual modules.

1.5. References

number	document	version
1	AUTOSAR_SWS_CryptoServiceManager.pdf	4.4.0, R19-11
2	AUTOSAR_SWS_CryptoInterface.pdf	4.4.0
3	AUTOSAR_SWS_CryptoDriver.pdf	4.4.0
4	[HSM_2.0spec] HSM_Framework_UserManual_Secure_Application_Guide_KOR_v2.0.pdf	2.00
5	[HSM_2.0spec] HSM_Framework_UserManual_Crypto_Service_KOR_v2.0.pdf	1.02
6	User's Manual of Hyundai AutoEver Cryptography Library.pdf	1.7.2

	Document No.	AUTOSAR HaeModule Crypto Driver User Manual	Revision Date	'24. 04
	-		Revision No.	1.03b

2. HaeModule Crypto Driver Overview

2.1. HaeModule Crypto Driver Architecture

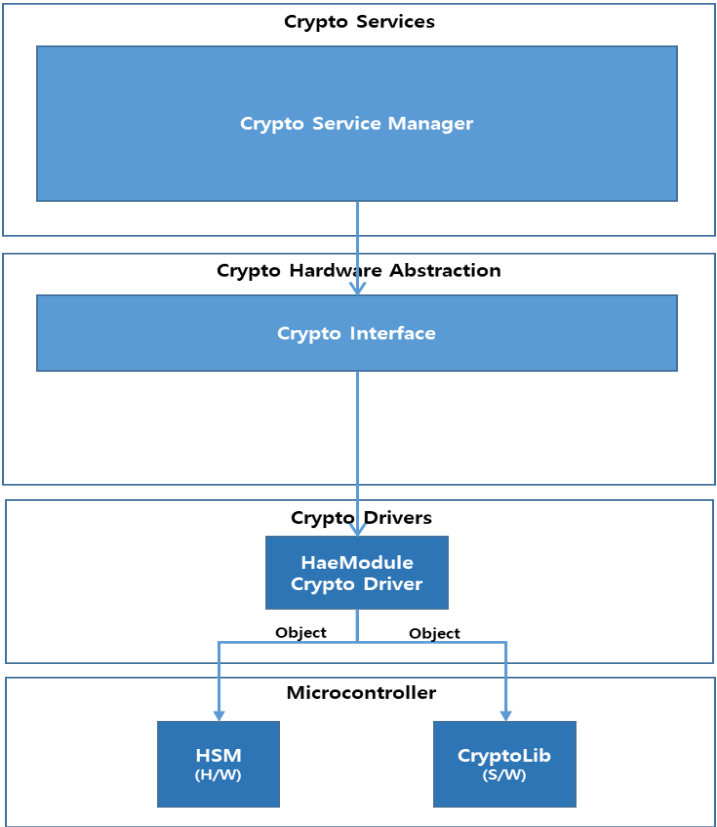


Figure 1. AUTOSAR Layered View with HaeModule Crypto Driver

In AUTOSAR, the Crypto Driver provides synchronous and asynchronous cryptographic services and key management services for them. The figure above shows the overall Crypto Stack architecture from the perspective of HaeModule Crypto Driver. HaeModule Crypto Driver provides services for HSMs and CryptoLib modules in the Crypto Drivers Layer .

Specifically, an HSM consists of an HSM driver and an HSM that runs on a separate core within the microcontroller. CryptoLib is available in the form of a library that provides cryptographic algorithm services implemented in software. For detailed structure and operation, please refer to the user manual provided for each module.

	Document No.	AUTOSAR HaeModule Crypto Driver User Manual	Revision Date	'24. 04
	-		Revision No.	1.03b

2.2. HaeModule Crypto Driver Structure

HaeModule Crypto shows the process of creating a Basic Software Module Description (BSWMD).

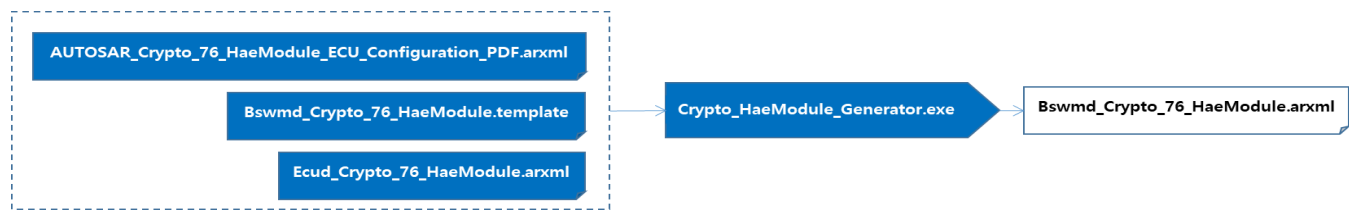


Figure 2. Create HaeModule Crypto BSWMD

It shows the process of creating a configuration file of the HaeModule Crypto Driver using Generator.

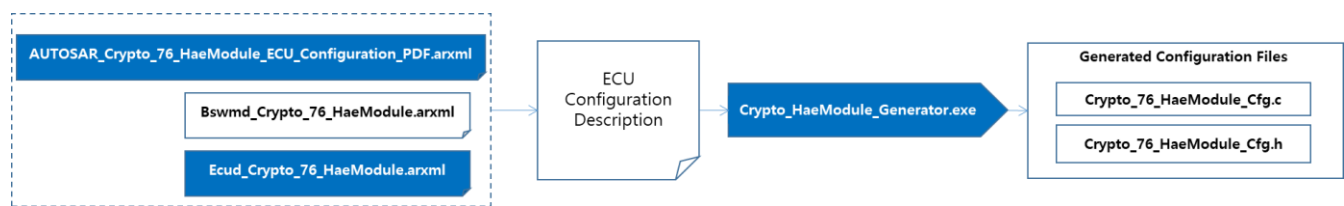


Figure 3. HSM Generated Configuration Files

Below is an overview of the source structure of the HaeModule Crypto Driver. **The HaeModule Crypto Driver runs on top of the HSM Driver or CryptoLib, so the HSM Driver Library or CryptoLib Library must be required.**

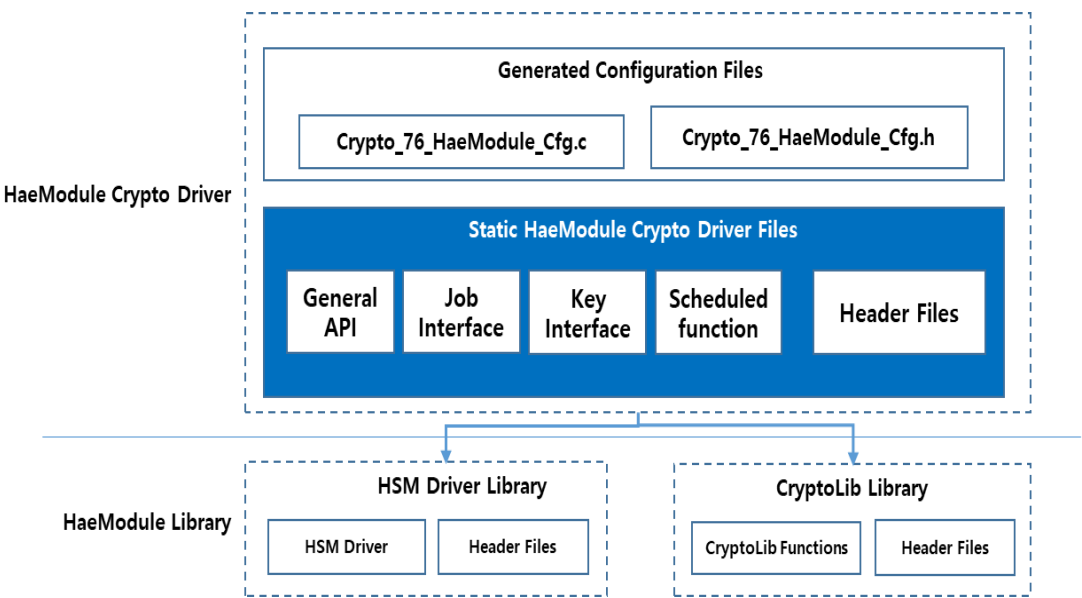



Figure 4. HaeModule Crypto Driver Source Structure

	Document No.	AUTOSAR HaeModule Crypto Driver User Manual	Revision Date	'24. 04
	-		Revision No.	1.03b

2.3. Files & Tools Provided

It provides the following files and tools:

It is provided under the 'Crypto_76_HaeModule_R44' package name, and the directory location for each provided item is as follows.

NO	item	location
1	Static HaeModule Crypto Driver Source File	delivery/src
2	Static HaeModule Crypto Driver Header File	delivery/inc
3	AUTOSAR_Crypto_76_HaeModule_ECU_Configuration_PDF.arxml	generator
4	Ecud_Crypto_76_HaeModule.arxml	generator
5	Bswmd_Crypto_76_HaeModule.template	generator
6	Crypto Generator Tools - Crypto_HaeModule_Generator.exe	generator
7	Crypto Generator Tools - Crypto_HaeModule_Generator.bat	generator
8	This Manual	doc

2.3.1. Static HaeModule Crypto Driver Source Files and Header Files

It provides a Crypto Driver function that meets the AUTOSAR specification. It is configured by the source file generated by the Crypto Generator tool. Call the HSM Driver or the CryptoLib API to perform the service.

2.3.2. AUTOSAR_Crypto_76_HaeModule_ECU_Configuration_PDF.arxml

Ecud_Crypto_76_HaeModule.arxml provides a schema for validation.

2.3.3. Ecud_Crypto_76_HaeModule.arxml

It follows the AUTOSAR XML format. It provides the ECU Description of the HaeModule Crypto Driver. This must be set in the input parameters of the Crypto Generator tool to create the configuration file and BSWMD file.

2.3.4. Bswmd_Crypto_76_HaeModule.template

Bswmd_Crypto_76_HaeModule.This is the template file for the arxml file.

2.3.5. Crypto Generator Tools

For more information, see “4. Crypto Generator Tools”See.

	Document No.	AUTOSAR HaeModule Crypto Driver User Manual	Revision Date	'24. 04
	-		Revision No.	1.03b

3. General Specifications

By default, it follows the AUTOSAR Classic Release 4.4.0 specification. However, for some specifications, the higher version is applied.

3.1. Operating Conditions

- The AUTOSAR Crypto Driver has been developed to support **the HSM security module and CryptoLib security module** provided by the company.
- Therefore, Crypto Driver functionality may be limited by the HSM security module or the CryptoLib security module provided by the company.
 - The CryptoLib version provides the Crypto Driver from v1.7.2.
- Although the Crypto Driver follows the AUTOSAR standard, there may be slight differences in the functionality or behavior of the company's HSM security module or CryptoLib security module. Therefore, please be sure to check the relevant function descriptions in this manual before use.
- Before the Crypto Driver can run, the company's HSM module or CryptoLib module must be working properly.
- The initialization function (HSM_DriverInitialize) of the HSM module must be executed and initialized after the initialization is completed, and the Crypto Driver initialization function (Crypto_76_HaeModule_Init) must be executed thereafter.
- The initialization function of the CryptoLib must be executed and the initialization must be completed after the initialization is completed, and the Crypto Driver initialization function must be executed and initialized after that.
- The Crypto Driver is not involved in the configuration of the HSM security module and the CryptoLib security module. It only uses the HSM Driver API or the CryptoLib API to provide the AUTOSAR standard Crypto Driver functionality. Therefore, in order for the Crypto Driver to work properly, the user must install the HSM security module and the CryptoLib security module separately from the Crypto Driver by referring to the user manual of each module.
- For definitions of user input used by the Crypto Driver, see Crypto_76_HaeModule_UserTypes.h.

3.2. Notice

- When using the SYNC mode HSM API, a watchdog timeout may occur in a HOST core user environment if the function return response time is longer than the cycle of the task.**

	Document No.	AUTOSAR HaeModule Crypto Driver User Manual	Revision Date	'24. 04
	-		Revision No.	1.03b

- Depending on the HSM API or MCU specification, it can take more than 100 ms of work.
- Since the return time is different for each user environment, you **need to measure it yourself and consider the task design.**(You are the one who designs the task so that it can work without any problems with the HSM API calls, which take a long time.)

3.3. Interface Support

In the following table, the Interface support of the Crypto Driver is indicated.

No.	Service name	Description	Support
1	Crypto_Init	Initializes the Crypto Driver.	○
2	Crypto_GetVersionInfo	Returns the version information of this module.	○
3	Crypto_ProcessJob	Performs the crypto primitive, that is configured in the job parameter	○
4	Crypto_CancelJob	This interface removes the provided job from the queue and cancels the processing of the job if possible	○
5	Crypto_KeyElementSet	Sets the given key element bytes to the key identified by cryptoKeyld.	○
6	Crypto_KeySetValid	Sets the key state of the key identified by cryptoKeyld to valid.	○
7	Crypto_KeyElementGet	This interface shall be used to get a key element of the key identified by the cryptoKeyld and store the key element in the memory location pointed by the result pointer. Note: If the actual key element is directly mapped to flash memory, there could be a bigger delay when calling this function (synchronous operation).	○
8	Crypto_KeyElementCopy	Copies a key element to another key element in the same crypto driver. Note: If the actual key element is directly mapped to flash memory, there could be a bigger delay when calling this function (synchronous operation)	○
9	Crypto_KeyElementCopyPartial	Copies a key element to another key element in the same crypto driver. The keyElementSourceOffset and keyElementCopyLength allows to copy just a part of the source key element into the destination. The offset of the target key is also specified with this function. Note: If the actual key element is directly mapped to flash memory, there could be a bigger delay when calling this function (synchronous operation).	○

	Document No.	AUTOSAR HaeModule Crypto Driver User Manual	Revision Date	'24. 04
	-		Revision No.	1.03b

10	Crypto_KeyCopy	Copies a key with all its elements to another key in the same crypto driver. Note: If the actual key element is directly mapped to flash memory, there could be a bigger delay when calling this function (synchronous operation)	○
11	Crypto_KeyElementIdsGet	Used to retrieve information which key elements are available in a given key.	○
12	Crypto_RandomSeed	This function generates the internal seed state using the provided entropy source. Furthermore, this function can be used to update the seed state with new entropy	○
13	Crypto_KeyGenerate	Generates new key material store it in the key identified by cryptoKeyld.	×
14	Crypto_KeyDerive	Derives a new key by using the key elements in the given key identified by the cryptoKeyld. The given key contains the key elements for the password, salt. The derived key is stored in the key element with the id 1 of the key identified by targetCryptoKeyld. The number of iterations is given in the key element CRYPTO_KE_KEYDERIVATION_ITERATIONS.	○
15	Crypto_KeyExchangeCalcPubVal	Calculates the public value for the key exchange and stores the public key in the memory location pointed by the public value pointer.	○
16	Crypto_KeyExchangeCalcSecret	Calculates the shared secret key for the key exchange with the key material of the key identified by the cryptoKeyld and the partner public key. The shared secret key is stored as a key element in the same key.	○
17	Crypto_CertificateParse	Parses the certificate data stored in the key element CRYPTO_KE_CERT_DATA and fills the key elements CRYPTO_KE_CERT_SIGNEDDATA, CRYPTO_KE_CERT_PARSEDPUBLICKEY and CRYPTO_KE_CERT_SIGNATURE.	×
18	Crypto_CertificateVerify	Verifies the certificate stored in the key referenced by cryptoValidateKeyld with the certificate stored in the key referenced by cryptoKeyld.	×
19	Crypto_MainFunction	If asynchronous job processing is configured and there are job queues, the function is called cyclically to process queued jobs.	○

Table 1. Crypto Driver Interface Support

	Document No.	AUTOSAR HaeModule Crypto Driver User Manual	Revision Date	'24. 04
	-		Revision No.	1.03b

3.4. What applies to AUTOSAR Classic Release R19-11

- As the following certificate interfaces have been deleted from the Key Management Interface, they are no longer supported by HaeModule Crypto Driver.
 - Crypto_CertificateParse
 - Crypto_CertificateVerify
- In the Crypto_JobType, the "CryptoKeyId" and "targetCryptoKeyId" variables are clearly defined.

	Document No.	AUTOSAR HaeModule Crypto Driver User Manual	Revision Date	'24. 04
	-		Revision No.	1.03b

4. Crypto Generator Tools

The Generator tool is a console application for Windows. Analyze the AUTOSAR ARXML file to generate the Crypto Configuration source file and the BSWMD file.

4.1 Usage and input parameters

If you run the Generator tool without input, you can get instructions on how to use it and input parameters.

4.2 Generate Source and Header Files

"Crypto_76_HaeModule_Cfg.c" and "Crypto_76_HaeModule_Cfg.h" file. **If you enter a Crylf ECUD file, you can reduce the size by generating a code for only the Object and Key connected to Crylf.** If you don't enter it, it will generate code for everything that is set in the Crypto ECUD. If you want the header file and the source file creation directory location to be in the same place, set "--ODirH" and "--OdirC" to be the same. Here's an example of how to use it:

e.g. `Crypto_HaeModule_Generator.exe --CODE --IEcudCrylf Ecud_Crylf.arxml --IEcudCrypto Ecud_Crypto_76_HaeModule.arxml --IBswmdCrypto Bswmd_Crypto_76_HaeModule.arxml --ODirH C:\W Generated\Wsw_Output\Winc --OdirC C:\WGenerated\Wsw_Output\Wsrc`

4.3 Creating a BSWMD File

Create a "Bswmd_Crypto_76_HaeModule.arxml" file. Here's an example:

e.g. `Crypto_HaeModule_Generator.exe --BSWMD --IEcudCrypto Ecud_Crypto_76_HaeModule.arxml --IBswmdCrypto Bswmd_Crypto_76_HaeModule.arxml --ODir C:\W Generated\Wsw_Output\Wbswmd`

	Document No.	AUTOSAR HaeModule Crypto Driver User Manual	Revision Date	'24. 04
	-		Revision No.	1.03b

5. Crypto Driver Objects

5.1. Support Modules

The HaeModule Crypto Driver has two objects to support its HSM module and its CryptoLib module.

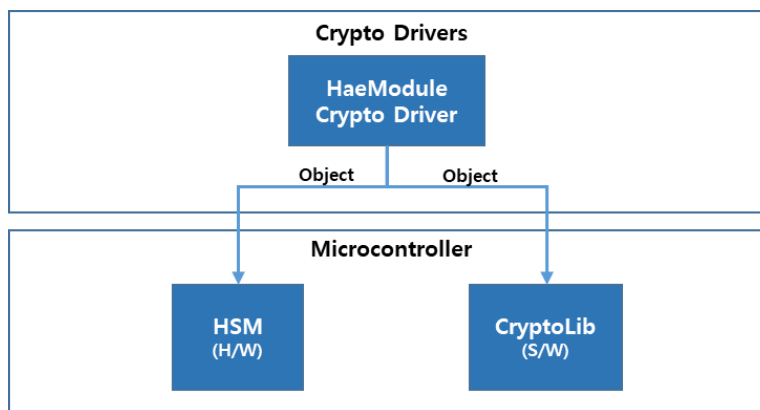


Figure 5. HaeModule Crypto Driver Supported Modules

5.2. Module-specific Objects and Primitives

The following are the Objects and Primitives of each supported module.

No.	Module	Object	Primitives
1	HSM Modules	HaeHsm	HaeHsmPrimitives
2	CryptoLib Modules	HaeCryptoLib	HaeCryptoLibPrimitives

Table 2. Module-specific Objects and Primitives

Set them in the object or Primitives of that name , respectively.

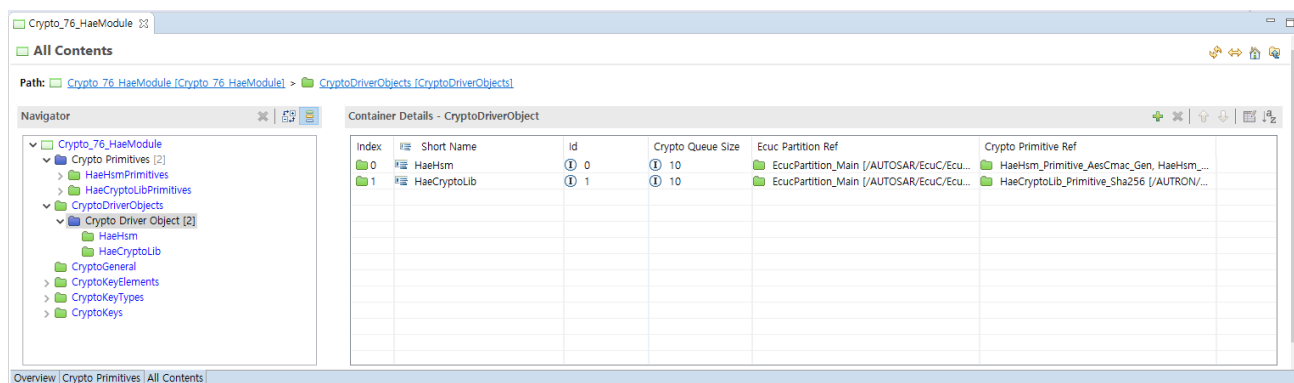


Figure 6. Module-specific Objects and Primitives

	Document No.	AUTOSAR HaeModule Crypto Driver User Manual	Revision Date	'24. 04
	-		Revision No.	1.03b

5.3. Module-specific prefix

We recommend using the following prefixes to differentiate between module-specific configurations or primitives. The supplied sources and ARXML follow this rule.

No.	Module	Prefix
1	HSM Modules	HaeHsm
2	CryptoLib Modules	HaeCryptoLib

Table 3. Module-specific prefix

If this is common between HSMs and CryptoLib modules, use a prefix called "HaeModule".

	Document No.	AUTOSAR HaeModule Crypto Driver User Manual	Revision Date	'24. 04
	-		Revision No.	1.03b

6. Job Processing

6.1. Job Primitive Settings by Crypto Algorithm

In this chapter, we will explain how to set up a job to use the Crypto algorithm. Job settings allow you to use the Crypto algorithm provided by HAE HSMs.

The 'Crypto_PrimitiveInfoType' table represents the values set in the 'Ecud_Crypto_76_HaeModule.arxml' file provided by default. Users can change the values to match the Crypto Service Manager (CSM) to suit their environment.

If the algorithm provided by HaeHsmPrimitives and HaeCryptoLibPrimitives is different, it is classified as "< >" .

6.1.1. CRYPTO_HASH

Crypto_PrimitiveInfoType

Service	Algorithm Family	Algorithm Secondary Family	Algorithm Mode
CRYPTO_HASH	CRYPTO_ALGOFAM_SHA1	CRYPTO_ALGOFAM_NOT_SET	CRYPTO_ALGOMODE_NOT_SET
CRYPTO_HASH	CRYPTO_ALGOFAM_SHA2_224	CRYPTO_ALGOFAM_NOT_SET	CRYPTO_ALGOMODE_NOT_SET
CRYPTO_HASH	CRYPTO_ALGOFAM_SHA2_256	CRYPTO_ALGOFAM_NOT_SET	CRYPTO_ALGOMODE_NOT_SET
CRYPTO_HASH	CRYPTO_ALGOFAM_SHA2_384	CRYPTO_ALGOFAM_NOT_SET	CRYPTO_ALGOMODE_NOT_SET
CRYPTO_HASH	CRYPTO_ALGOFAM_SHA2_512	CRYPTO_ALGOFAM_NOT_SET	CRYPTO_ALGOMODE_NOT_SET
CRYPTO_HASH	CRYPTO_ALGOFAM_SHA2_512_224	CRYPTO_ALGOFAM_NOT_SET	CRYPTO_ALGOMODE_NOT_SET
CRYPTO_HASH	CRYPTO_ALGOFAM_SHA2_512_256	CRYPTO_ALGOFAM_NOT_SET	CRYPTO_ALGOMODE_NOT_SET
CRYPTO_HASH	CRYPTO_ALGOFAM_SHAKE256	CRYPTO_ALGOFAM_NOT_SET	CRYPTO_ALGOMODE_NOT_SET

Crypto_JobPrimitiveInputOutputType

Type	Parameter	Description
const uint8*	inputPtr	Input data
uint32	inputLength	inputPtr byte size
uint8*	outputPtr	Hash Result Data
uint32*	outputLengthPtr	outputPtr byte size

	Document No.	AUTOSAR HaeModule Crypto Driver User Manual	Revision Date	'24. 04
	-		Revision No.	1.03b

6.1.2. CRYPTO_MACGENERATE

6.1.2.1. TDES CMAC

Crypto_PrimitiveInfoType < [HaeCryptoLibPrimitives](#) >

Service	Algorithm Family	Algorithm Secondary Family	Algorithm Mode
CRYPTO_MACGENERATE	CRYPTO_ALGOFAM_TDES	CRYPTO_ALGOFAM_NOT_SET	CRYPTO_ALGOMODE_CMACE

Crypto_JobPrimitiveInputOutputType

Type	Parameter	Description
const uint8*	inputPtr	Input data to compute in CMAC
uint32	inputLength	The byte size of the input data to be computed by CMAC
uint8*	outputPtr	CMAC Result Data
uint32*	outputLengthPtr	outputPtr byte size

Crypto_JobType

Type	Parameter	Description
uint32	cryptoKeyId	TDES CMAC Key Number

- The cryptoKey must have a valid value set in the following Key Element before the service can be executed.

key element Name	key element ID	Comments
CRYPTO_KE_MAC_KEY	1	TDES CMAC Key Value (8-byte)

	Document No.	AUTOSAR HaeModule Crypto Driver User Manual	Revision Date	'24. 04
	-		Revision No.	1.03b

6.1.2.2. AES CMAC

Crypto_PrimitiveInfoType

Service	Algorithm Family	Algorithm Secondary Family	Algorithm Mode
CRYPTO_MACGENERATE	CRYPTO_ALGOFAM_AES	CRYPTO_ALGOFAM_NOT_SET	CRYPTO_ALGOMODE_CMAC

Crypto_JobPrimitiveInputOutputType

Type	Parameter	Description
const uint8*	inputPtr	Input data to compute in CMAC
uint32	inputLength	The byte size of the input data to be computed by CMAC
uint8*	outputPtr	CMAC Result Data
uint32*	outputLengthPtr	outputPtr byte size

Crypto_AlgorithmInfoType

Type	Parameter	Description
uint32	keyLength	The key length in bytes to be used with that algorithm Applicable AES Key Byte Length <ul style="list-style-type: none"> AES-128 : 16 AES-192: 24 (not allowed by HaeHsmPrimitives) AES-256: 32 (not allowed by HaeHsmPrimitives)

- The application algorithm is determined by the setting value.

Crypto_JobType

Type	Parameter	Description
uint32	cryptoKeyld	AES CMAC Key Number

- The cryptoKey must have a valid value set in the following Key Element before the service can be executed.

key element Name	key element ID	Comments
CRYPTO_KE_MAC_KEY	1	AES CMAC Key Values

	Document No.	AUTOSAR HaeModule Crypto Driver User Manual	Revision Date	'24. 04
	-		Revision No.	1.03b

6.1.2.3. AES GMAC

Crypto_PrimitiveInfoType < HaeCryptoLibPrimitives >

Service	Algorithm Family	Algorithm Secondary Family	Algorithm Mode
CRYPTO_MACGENERATE	CRYPTO_ALGOFAM_AES	CRYPTO_ALGOFAM_NOT_SET	CRYPTO_ALGOMODE_GMAC

Crypto_JobPrimitiveInputOutputType

Type	Parameter	Description
const uint8*	inputPtr	Input message data to compute in GMAC (AAD is not used)
uint32	inputLength	The byte size of the input message data to be computed by GMAC
uint8*	outputPtr	GMAC Result Data
uint32*	outputLengthPtr	outputPtr byte size

- When using CRYPTO_OPERATIONMODE_SINGLECALL , g_cryptoLib_gmacLastUpdateCall = 1 is required
- In other modes, when entering a data block, g_cryptoLib_gmacLastUpdateCall = 0 and inputPtr can only be entered in multiples of 16 bytes.
When entering the last block of data, after setting g_cryptoLib_gmacLastUpdateCall = 1, it can be anything other than 16 bytes in length.
For additional information, please refer to the CryptoLib manual.

Crypto_AlgorithmInfoType

Type	Parameter	Description
uint32	keyLength	The key length in bytes to be used with that algorithm Applicable AES Key Byte Length <ul style="list-style-type: none"> • AES-128 : 16 • AES-192 : 24 • AES-256 : 32

- The application algorithm is determined by the setting value.

Crypto_JobType

Type	Parameter	Description
uint32	cryptoKeyId	AES GMAC Key Number

- The cryptoKey must have a valid value set in the following Key Element before the service can be executed.

key element Name	key element ID	Comments
CRYPTO_KE_MAC_KEY	1	AES GMAC Key Values
CRYPTO_KE_MAC_IV	1005	AES GMAC IV Values

	Document No.	AUTOSAR HaeModule Crypto Driver User Manual	Revision Date	'24. 04
	-		Revision No.	1.03b

	Document No.	AUTOSAR HaeModule Crypto Driver User Manual	Revision Date	'24. 04
	-		Revision No.	1.03b

6.1.2.4. HMAC

Crypto_PrimitiveInfoType < HaeHsmPrimitives >

Service	Algorithm Family	Algorithm Secondary Family	Algorithm Mode
CRYPTO_MACGENERATE	CRYPTO_ALGOFAM_SHA2_256	CRYPTO_ALGOFAM_NOT_SET	CRYPTO_ALGOMODE_HMAC

Crypto_PrimitiveInfoType < HaeCryptoLibPrimitives >

Service	Algorithm Family	Algorithm Secondary Family	Algorithm Mode
CRYPTO_MACGENERATE	CRYPTO_ALGOFAM_SHA1	CRYPTO_ALGOFAM_NOT_SET	CRYPTO_ALGOMODE_HMAC
CRYPTO_MACGENERATE	CRYPTO_ALGOFAM_SHA2_224	CRYPTO_ALGOFAM_NOT_SET	CRYPTO_ALGOMODE_HMAC
CRYPTO_MACGENERATE	CRYPTO_ALGOFAM_SHA2_256	CRYPTO_ALGOFAM_NOT_SET	CRYPTO_ALGOMODE_HMAC
CRYPTO_MACGENERATE	CRYPTO_ALGOFAM_SHA2_384	CRYPTO_ALGOFAM_NOT_SET	CRYPTO_ALGOMODE_HMAC
CRYPTO_MACGENERATE	CRYPTO_ALGOFAM_SHA2_512	CRYPTO_ALGOFAM_NOT_SET	CRYPTO_ALGOMODE_HMAC

Crypto_JobPrimitiveInputOutputType

Type	Parameter	Description
const uint8*	inputPtr	Input data to compute in the HMAC
uint32	inputLength	The byte size of the input data to be computed by the HMAC
uint8*	outputPtr	HMAC Result Data
uint32*	outputLengthPtr	outputPtr byte size

Crypto_JobType

Type	Parameter	Description
uint32	cryptoKeyId	HMAC Key Number

- The cryptoKey must have a valid value set in the following Key Element before the service can be executed.

key element Name	key element ID	Comments
CRYPTO_KE_MAC_KEY	1	HMAC Key Values

	Document No.	AUTOSAR HaeModule Crypto Driver User Manual	Revision Date	'24. 04
	-		Revision No.	1.03b

6.1.3. CRYPTO_MACVERIFY

6.1.3.1. TDES CMAC

Crypto_PrimitiveInfoType < HaeCryptoLibPrimitives >

Service	Algorithm Family	Algorithm Secondary Family	Algorithm Mode
CRYPTO_MACVERIFY	CRYPTO_ALGOFAM_TDES	CRYPTO_ALGOFAM_NOT_SET	CRYPTO_ALGOMODE_CMIC

Crypto_JobPrimitiveInputOutputType

Type	Parameter	Description
const uint8*	inputPtr	Holds a pointer to the data for which the MAC shall be verified.
uint32	inputLength	Contains the number of data bytes for which the MAC shall be verified.
const uint8*	secondaryInputPtr	Holds a pointer to the MAC to be verified.
uint32	secondaryInputLength	Contains the MAC length in BITS to be verified.
Crypto_VerifyResultType*	verifyPtr	Holds a pointer to the memory location, which will hold the result of the MAC verification.

Crypto_JobType

Type	Parameter	Description
uint32	cryptoKeyId	TDES CMAC Key Number

- The cryptoKey must have a valid value set in the following Key Element before the service can be executed.

key element Name	key element ID	Comments
CRYPTO_KE_MAC_KEY	1	TDES CMAC Key Value (8-byte)

	Document No.	AUTOSAR HaeModule Crypto Driver User Manual	Revision Date	'24. 04
	-		Revision No.	1.03b

6.1.3.2. AES CMAC

Crypto_PrimitiveInfoType

Service	Algorithm Family	Algorithm Secondary Family	Algorithm Mode
CRYPTO_MACVERIFY	CRYPTO_ALGOFAM_AES	CRYPTO_ALGOFAM_NOT_SET	CRYPTO_ALGOMODE_CMACE

Crypto_JobPrimitiveInputOutputType

Type	Parameter	Description
const uint8*	inputPtr	Holds a pointer to the data for which the MAC shall be verified.
uint32	inputLength	Contains the number of data bytes for which the MAC shall be verified.
const uint8*	secondaryInputPtr	Holds a pointer to the MAC to be verified.
uint32	secondaryInputLength	Contains the MAC length in BITS to be verified.
Crypto_VerifyResultType*	verifyPtr	Holds a pointer to the memory location, which will hold the result of the MAC verification.

Crypto_AlgorithmInfoType

Type	Parameter	Description
uint32	keyLength	The key length in bytes to be used with that algorithm Applicable AES Key Byte Length <ul style="list-style-type: none"> AES-128 : 16 AES-192: 24 (not allowed by HaeHsmPrimitives) AES-256: 32 (not allowed by HaeHsmPrimitives)

- The application algorithm is determined by the setting value.

Crypto_JobType

Type	Parameter	Description
uint32	cryptoKeyId	AES CMAC Key Number

- The cryptoKey must have a valid value set in the following Key Element before the service can be executed.

key element Name	key element ID	Comments
CRYPTO_KE_MAC_KEY	1	AES CMAC Key Values

	Document No.	AUTOSAR HaeModule Crypto Driver User Manual	Revision Date	'24. 04
	-		Revision No.	1.03b

6.1.3.3. AES GMAC

Crypto_PrimitiveInfoType < HaeCryptoLibPrimitives >

Service	Algorithm Family	Algorithm Secondary Family	Algorithm Mode
CRYPTO_MACVERIFY	CRYPTO_ALGOFAM_AES	CRYPTO_ALGOFAM_NOT_SET	CRYPTO_ALGOMODE_GMAC

Crypto_JobPrimitiveInputOutputType

Type	Parameter	Description
const uint8*	inputPtr	Holds a pointer to the data for which the MAC shall be verified. (not used AAD)
uint32	inputLength	Contains the number of data bytes for which the MAC shall be verified.
const uint8*	secondaryInputPtr	Holds a pointer to the MAC to be verified.
uint32	secondaryInputLength	Contains the MAC length in BITS to be verified.
Crypto_VerifyResultType*	verifyPtr	Holds a pointer to the memory location, which will hold the result of the MAC verification.


- When using CRYPTO_OPERATIONMODE_SINGLECALL, g_cryptoLib_gmacLastUpdateCall = 1 is required
- In other modes, when entering a data block, inputPtr can only be entered in multiples of 16 bytes after setting g_cryptoLib_gmacLastUpdateCall = 0. When entering the last block of data, after setting g_cryptoLib_gmacLastUpdateCall = 1, it can be anything other than 16 bytes in length. For additional information, please refer to the CryptoLib manual.

Crypto_JobType

Type	Parameter	Description
uint32	cryptoKeyId	AES GMAC Key Number

- The cryptoKey must have a valid value set in the following Key Element before the service can be executed.

key element Name	key element ID	Comments
CRYPTO_KE_MAC_KEY	1	AES GMAC Key Values
CRYPTO_KE_MAC_IV	1005	AES GMAC Key Values

	Document No.	AUTOSAR HaeModule Crypto Driver User Manual	Revision Date	'24. 04
	-		Revision No.	1.03b

6.1.3.4. HMAC

Crypto_PrimitiveInfoType < HaeHsmPrimitives >

Service	Algorithm Family	Algorithm Secondary Family	Algorithm Mode
CRYPTO_MACVERIFY	CRYPTO_ALGOFAM_SHA2_256	CRYPTO_ALGOFAM_NOT_SET	CRYPTO_ALGOMODE_HMAC

Crypto_PrimitiveInfoType < HaeCryptoLibPrimitives >

Service	Algorithm Family	Algorithm Secondary Family	Algorithm Mode
CRYPTO_MACVERIFY	CRYPTO_ALGOFAM_SHA1	CRYPTO_ALGOFAM_NOT_SET	CRYPTO_ALGOMODE_HMAC
CRYPTO_MACVERIFY	CRYPTO_ALGOFAM_SHA2_224	CRYPTO_ALGOFAM_NOT_SET	CRYPTO_ALGOMODE_HMAC
CRYPTO_MACVERIFY	CRYPTO_ALGOFAM_SHA2_256	CRYPTO_ALGOFAM_NOT_SET	CRYPTO_ALGOMODE_HMAC
CRYPTO_MACVERIFY	CRYPTO_ALGOFAM_SHA2_384	CRYPTO_ALGOFAM_NOT_SET	CRYPTO_ALGOMODE_HMAC
CRYPTO_MACVERIFY	CRYPTO_ALGOFAM_SHA2_512	CRYPTO_ALGOFAM_NOT_SET	CRYPTO_ALGOMODE_HMAC

Crypto_JobPrimitiveInputOutputType

Type	Parameter	Description
const uint8*	inputPtr	Holds a pointer to the data for which the MAC shall be verified.
uint32	inputLength	Contains the number of data bytes for which the MAC shall be verified.
const uint8*	secondaryInputPtr	Holds a pointer to the MAC to be verified.
uint32	secondaryInputLength	Contains the MAC length in BITS to be verified.
Crypto_VerifyResultType*	verifyPtr	Holds a pointer to the memory location, which will hold the result of the MAC verification.

Crypto_JobType

Type	Parameter	Description
uint32	cryptoKeyId	HMAC Key Number

- The cryptoKey must have a valid value set in the following Key Element before the service can be executed.

key element Name	key element ID	Comments
CRYPTO_KE_MAC_KEY	1	HMAC Key Values

	Document No.	AUTOSAR HaeModule Crypto Driver User Manual	Revision Date	'24. 04
	-		Revision No.	1.03b

6.1.4. CRYPTO_ENCRYPT

6.1.4.1. TDES

Crypto_PrimitiveInfoType < [HaeCryptoLibPrimitives](#) >

Service	Algorithm Family	Algorithm Secondary Family	Algorithm Mode
CRYPTO_ENCRYPT	CRYPTO_ALGOFAM_TDES	CRYPTO_ALGOFAM_NOT_SET	CRYPTO_ALGOMODE_ECB
CRYPTO_ENCRYPT	CRYPTO_ALGOFAM_TDES	CRYPTO_ALGOFAM_NOT_SET	CRYPTO_ALGOMODE_CBC
CRYPTO_ENCRYPT	CRYPTO_ALGOFAM_TDES	CRYPTO_ALGOFAM_NOT_SET	CRYPTO_ALGOMODE_CTR

Crypto_JobPrimitiveInputOutputType

Type	Parameter	Description
const uint8*	inputPtr	Plaintext Data Pointer
uint32	inputLength	Plaintext Data Byte Size
uint8*	outputPtr	Encrypted data storage pointer
uint32*	outputLengthPtr	Encrypted data length pointer

Crypto_JobType

Type	Parameter	Description
uint32	cryptoKeyId	TDES Key Number

- The cryptoKey must have a valid value set in the following Key Element before the service can be executed.

key element Name	key element ID	Comments
CRYPTO_KE_CIPHER_KEY	1	TDES Key Value (8-byte)
CRYPTO_KE_CIPHER_IV	5	TDES IV value (8-byte, for ECB, not required)

	Document No.	AUTOSAR HaeModule Crypto Driver User Manual	Revision Date	'24. 04
	-		Revision No.	1.03b

6.1.4.2. AES

Crypto_PrimitiveInfoType

Service	Algorithm Family	Algorithm Secondary Family	Algorithm Mode
CRYPTO_ENCRYPT	CRYPTO_ALGOFAM_AES	CRYPTO_ALGOFAM_NOT_SET	CRYPTO_ALGOMODE_ECB
CRYPTO_ENCRYPT	CRYPTO_ALGOFAM_AES	CRYPTO_ALGOFAM_NOT_SET	CRYPTO_ALGOMODE_CBC
CRYPTO_ENCRYPT	CRYPTO_ALGOFAM_AES	CRYPTO_ALGOFAM_NOT_SET	CRYPTO_ALGOMODE_CTR
CRYPTO_ENCRYPT	CRYPTO_ALGOFAM_AES	CRYPTO_ALGOFAM_NOT_SET	CRYPTO_ALGOMODE_OFB

Crypto_JobPrimitiveInputOutputType

Type	Parameter	Description
const uint8*	inputPtr	Plaintext Data Pointer
uint32	inputLength	Plaintext Data Byte Size
uint8*	outputPtr	Encrypted data storage pointer
uint32*	outputLengthPtr	Encrypted data length pointer

Crypto_AlgorithmInfoType

Type	Parameter	Description
uint32	keyLength	The key length in bytes to be used with that algorithm Applicable AES Key Byte Length <ul style="list-style-type: none"> AES-128 : 16 AES-192: 24 (not allowed by HaeHsmPrimitives) AES-256 : 32

- The application algorithm is determined by the setting value.

Crypto_JobType

Type	Parameter	Description
uint32	cryptoKeyId	AES Key Number

- The cryptoKey must have a valid value set in the following Key Element before the service can be executed.

key element Name	key element ID	Comments
CRYPTO_KE_CIPHER_KEY	1	AES GMAC Key Values
CRYPTO_KE_CIPHER_IV	5	AES GMAC IV value (not required for ECB)

	Document No.	AUTOSAR HaeModule Crypto Driver User Manual	Revision Date	'24. 04
	-		Revision No.	1.03b

6.1.4.3. CHACHA20

Crypto_PrimitiveInfoType < [HaeCryptoLibPrimitives](#) >

Service	Algorithm Family	Algorithm Secondary Family	Algorithm Mode
CRYPTO_ENCRYPT	CRYPTO_ALGOFAM_CHACHA	CRYPTO_ALGOFAM_NOT_SET	CRYPTO_ALGOMODE_20ROUNDS

Crypto_JobPrimitiveInputOutputType

Type	Parameter	Description
const uint8*	inputPtr	Plaintext Data Pointer
uint32	inputLength	Plaintext Data Byte Size
uint8*	outputPtr	Encrypted data storage pointer
uint32*	outputLengthPtr	Encrypted data length pointer

Crypto_JobType

Type	Parameter	Description
uint32	cryptoKeyId	CHACHA20 Key Number

- The cryptoKey must have a valid value set in the following Key Element before the service can be executed.

key element Name	key element ID	Comments
CRYPTO_KE_CIPHER_KEY	1	CHACHA20 Key Value (32-byte)
CRYPTO_KE_CIPHER_IV	5	CHACHA20 InitialCounter nonce (16-byte)

	Document No.	AUTOSAR HaeModule Crypto Driver User Manual	Revision Date	'24. 04
	-		Revision No.	1.03b

6.1.4.4. RSAES-PKCS1_v1_5

Crypto_PrimitiveInfoType

Service	Algorithm Family	Algorithm Secondary Family	Algorithm Mode
CRYPTO_ENCRYPT	CRYPTO_ALGOFAM_RSA	CRYPTO_ALGOFAM_NOT_SET	CRYPTO_ALGOMODE_RSAES_PKCS1_v1_5

Crypto_JobPrimitiveInputOutputType

Type	Parameter	Description
const uint8*	inputPtr	Plaintext Data Pointer
uint32	inputLength	Plain text data data byte size (up to 256 bytes in length)
uint8*	outputPtr	Encrypted data storage pointer
uint32*	outputLengthPtr	Encrypted data length pointer

Crypto_JobType

Type	Parameter	Description
uint32	cryptoKeyId	RSA Key Number

- The cryptoKey must have a valid value set in the following Key Element before the service can be executed.

key element Name	key element ID	Comments
CRYPTO_KE_CIPHER_KEY	1	RSA public key value (260-byte = N E (4-byte))

	Document No.	AUTOSAR HaeModule Crypto Driver User Manual	Revision Date	'24. 04
	-		Revision No.	1.03b

6.1.4.5. RSAES-OAEP

- CSM R4.4 CsmJobKeyDeriveAlgorithmSecondaryFamily not supported (R22-11 supported)

- After setting the CRYPTO_ALGOFAM_CUSTOM, enter a value in the AlgorithmSecondaryFamilyCustom

Crypto_PrimitiveInfoType < HaeHsmPrimitives >

Service	Algorithm Family	Algorithm Secondary Family	Algorithm Mode
CRYPTO_ENCRYPT	CRYPTO_ALGOFAM_RSA	CRYPTO_ALGOFAM_SHA2_256	CRYPTO_ALGOMODE_RSAES_OAEP

Crypto_PrimitiveInfoType < HaeCryptoLibPrimitives >

Service	Algorithm Family	Algorithm Secondary Family	Algorithm Mode
CRYPTO_ENCRYPT	CRYPTO_ALGOFAM_RSA	CRYPTO_ALGOFAM_SHA1	CRYPTO_ALGOMODE_RSAES_OAEP
CRYPTO_ENCRYPT	CRYPTO_ALGOFAM_RSA	CRYPTO_ALGOFAM_SHA2_224	CRYPTO_ALGOMODE_RSAES_OAEP
CRYPTO_ENCRYPT	CRYPTO_ALGOFAM_RSA	CRYPTO_ALGOFAM_SHA2_256	CRYPTO_ALGOMODE_RSAES_OAEP
CRYPTO_ENCRYPT	CRYPTO_ALGOFAM_RSA	CRYPTO_ALGOFAM_SHA2_384	CRYPTO_ALGOMODE_RSAES_OAEP
CRYPTO_ENCRYPT	CRYPTO_ALGOFAM_RSA	CRYPTO_ALGOFAM_SHA2_512	CRYPTO_ALGOMODE_RSAES_OAEP

Crypto_JobPrimitiveInputOutputType

Type	Parameter	Description
const uint8*	inputPtr	Pointer in HAEMODULE_RSAES_OAEP_INPUT_t
uint32	inputLength	inputPtr byte size
uint8*	outputPtr	Encrypted data storage pointer
uint32*	outputLengthPtr	Encrypted data length pointer

HAEMODULE_RSAES_OAEP_INPUT_t

Name	HSM_RSAES_OAEP_INPUT_t		
Type	Structure		
Element	uint8*	textPtr	Plaintext data
	uint32	textLength	Plaintext data length
	uint8*	labelPtr	Label Data
	uint32	labelLength	Length of Label Data
Description	About the input data of the algorithm		

Crypto_JobType

Type	Parameter	Description
uint32	cryptoKeyld	RSA Key Number

- The cryptoKey must have a valid value set in the following Key Element before the service can be executed.

key element Name	key element ID	Comments
CRYPTO_KE_CIPHER_KEY	1	RSA public key value (260-byte = N E (4-byte))

	Document No.	AUTOSAR HaeModule Crypto Driver User Manual	Revision Date	'24. 04
	-		Revision No.	1.03b

CRYPTO_KE_CIPHER_SEED	1016	RSAES-OAEP seed value (hash output length)
-----------------------	------	--

	Document No.	AUTOSAR HaeModule Crypto Driver User Manual	Revision Date	'24. 04
	-		Revision No.	1.03b

6.1.5. CRYPTO_DECRYPT

6.1.5.1. TDES

Crypto_PrimitiveInfoType < [HaeCryptoLibPrimitives](#) >

Service	Algorithm Family	Algorithm Secondary Family	Algorithm Mode
CRYPTO_DECRYPT	CRYPTO_ALGOFAM_TDES	CRYPTO_ALGOFAM_NOT_SET	CRYPTO_ALGOMODE_ECB
CRYPTO_DECRYPT	CRYPTO_ALGOFAM_TDES	CRYPTO_ALGOFAM_NOT_SET	CRYPTO_ALGOMODE_CBC
CRYPTO_DECRYPT	CRYPTO_ALGOFAM_TDES	CRYPTO_ALGOFAM_NOT_SET	CRYPTO_ALGOMODE_CTR

Crypto_JobPrimitiveInputOutputType


Type	Parameter	Description
const uint8*	inputPtr	Ciphertext Data Pointer
uint32	inputLength	Ciphertext Data Byte Size
uint8*	outputPtr	Decrypted data storage pointer
uint32*	outputLengthPtr	Decrypted data length pointer

Crypto_JobType

Type	Parameter	Description
uint32	cryptoKeyId	AES Key Number

- The cryptoKey must have a valid value set in the following Key Element before the service can be executed.

key element Name	key element ID	Comments
CRYPTO_KE_CIPHER_KEY	1	TDES Key Value (8-byte)
CRYPTO_KE_CIPHER_IV	5	TDES IV value (8-byte, for ECB, not required)

	Document No.	AUTOSAR HaeModule Crypto Driver User Manual	Revision Date	'24. 04
	-		Revision No.	1.03b

6.1.5.2. AES

Crypto_PrimitiveInfoType

Service	Algorithm Family	Algorithm Secondary Family	Algorithm Mode
CRYPTO_DECRYPT	CRYPTO_ALGOFAM_AES	CRYPTO_ALGOFAM_NOT_SET	CRYPTO_ALGOMODE_ECB
CRYPTO_DECRYPT	CRYPTO_ALGOFAM_AES	CRYPTO_ALGOFAM_NOT_SET	CRYPTO_ALGOMODE_CBC
CRYPTO_DECRYPT	CRYPTO_ALGOFAM_AES	CRYPTO_ALGOFAM_NOT_SET	CRYPTO_ALGOMODE_CTR
CRYPTO_DECRYPT	CRYPTO_ALGOFAM_AES	CRYPTO_ALGOFAM_NOT_SET	CRYPTO_ALGOMODE_OFB

Crypto_JobPrimitiveInputOutputType

Type	Parameter	Description
const uint8*	inputPtr	Ciphertext Data Pointer
uint32	inputLength	Ciphertext Data Byte Size
uint8*	outputPtr	Decrypted data storage pointer
uint32*	outputLengthPtr	Decrypted data length pointer

Crypto_AlgorithmInfoType

Type	Parameter	Description
uint32	keyLength	The key length in bytes to be used with that algorithm Applicable AES Key Byte Length <ul style="list-style-type: none"> AES-128 : 16 AES-192: 24 (not allowed by HaeHsmPrimitives) AES-256 : 32

- The application algorithm is determined by the setting value.

Crypto_JobType

Type	Parameter	Description
uint32	cryptoKeyId	AES Key Number

- The cryptoKey must have a valid value set in the following Key Element before the service can be executed.

key element Name	key element ID	Comments
CRYPTO_KE_CIPHER_KEY	1	AES Key Values
CRYPTO_KE_CIPHER_IV	5	AES IV value (for ECB, not required)

	Document No.	AUTOSAR HaeModule Crypto Driver User Manual	Revision Date	'24. 04
	-		Revision No.	1.03b

6.1.5.3. CHACHA20

Crypto_PrimitiveInfoType < [HaeCryptoLibPrimitives](#) >

Service	Algorithm Family	Algorithm Secondary Family	Algorithm Mode
CRYPTO_DECRYPT	CRYPTO_ALGOFAM_CHACHA	CRYPTO_ALGOFAM_NOT_SET	CRYPTO_ALGOMODE_20ROUNDS

Crypto_JobPrimitiveInputOutputType

Type	Parameter	Description
const uint8*	inputPtr	Ciphertext Data Pointer
uint32	inputLength	Ciphertext Data Byte Size
uint8*	outputPtr	Decrypted data storage pointer
uint32*	outputLengthPtr	Decrypted data length pointer

Crypto_JobType

Type	Parameter	Description
uint32	cryptoKeyId	CHACHA20 Key Number

- The cryptoKey must have a valid value set in the following Key Element before the service can be executed.

key element Name	key element ID	Comments
CRYPTO_KE_CIPHER_KEY	1	CHACHA20 Key Value (32-byte)
CRYPTO_KE_CIPHER_IV	5	CHACHA20 InitialCounter nonce (16-byte)

	Document No.	AUTOSAR HaeModule Crypto Driver User Manual	Revision Date	'24. 04
	-		Revision No.	1.03b

6.1.5.4. RSAES-PKCS1_v1_5

Crypto_PrimitiveInfoType

Service	Algorithm Family	Algorithm Secondary Family	Algorithm Mode
CRYPTO_ENCRYPT	CRYPTO_ALGOFAM_RSA	CRYPTO_ALGOFAM_NOT_SET	CRYPTO_ALGOMODE_RSAES_PKCS1_v1_5

Crypto_JobPrimitiveInputOutputType

Type	Parameter	Description
const uint8*	inputPtr	Password message pointer
uint32	inputLength	Cryptographic message data byte size (up to 256 bytes in length)
uint8*	outputPtr	Contains the pointer to the memory location where the decrypted data shall be stored.
uint32*	outputLengthPtr	Holds a pointer to the memory location in which the output length information is stored in bytes. On calling this function, this parameter shall contain the size of the buffer provided by outputPtr. When the request has finished, the actual length of the returned value shall be stored.

Crypto_JobType

Type	Parameter	Description
uint32	cryptoKeyld	RSA Key Number

- The cryptoKey must have a valid value set in the following Key Element before the service can be executed.

key element Name	key element ID	Comments
CRYPTO_KE_CIPHER_KEY	1	RSA private key value (512-byte = N D)

	Document No.	AUTOSAR HaeModule Crypto Driver User Manual	Revision Date	'24. 04
	-		Revision No.	1.03b

6.1.5.5. RSAES-OAEP

- CSM R4.4 CsmDecryptAlgorithmSecondaryFamily not supported (R22-11 supported)

Crypto_PrimitiveInfoType < HaeHsmPrimitives >

Service	Algorithm Family	Algorithm Secondary Family	Algorithm Mode
CRYPTO_DECRYPT	CRYPTO_ALGOFAM_RSA	CRYPTO_ALGOFAM_SHA2_256	CRYPTO_ALGOMODE_RSAES_OAEP

Crypto_PrimitiveInfoType < HaeCryptoLibPrimitives >

Service	Algorithm Family	Algorithm Secondary Family	Algorithm Mode
CRYPTO_DECRYPT	CRYPTO_ALGOFAM_RSA	CRYPTO_ALGOFAM_SHA1	CRYPTO_ALGOMODE_RSAES_OAEP
CRYPTO_DECRYPT	CRYPTO_ALGOFAM_RSA	CRYPTO_ALGOFAM_SHA2_224	CRYPTO_ALGOMODE_RSAES_OAEP
CRYPTO_DECRYPT	CRYPTO_ALGOFAM_RSA	CRYPTO_ALGOFAM_SHA2_256	CRYPTO_ALGOMODE_RSAES_OAEP
CRYPTO_DECRYPT	CRYPTO_ALGOFAM_RSA	CRYPTO_ALGOFAM_SHA2_384	CRYPTO_ALGOMODE_RSAES_OAEP
CRYPTO_DECRYPT	CRYPTO_ALGOFAM_RSA	CRYPTO_ALGOFAM_SHA2_512	CRYPTO_ALGOMODE_RSAES_OAEP

Crypto_JobPrimitiveInputOutputType

Type	Parameter	Description
const uint8*	inputPtr	Pointer in HAEMODULE_RSAES_OAEP_INPUT_t
uint32	inputLength	inputPtr byte size
uint8*	outputPtr	Decrypted data storage pointer
uint32*	outputLengthPtr	Decrypted data length pointer

HAEMODULE_RSAES_OAEP_INPUT_t

Name	HSM_RSAES_OAEP_INPUT_t		
Type	Structure		
Element	uint8*	textPtr	256 Bytes of ciphertext data
	uint32	textLength	Ciphertext data length
	uint8*	labelPtr	Label Data
	uint32	labelLength	Length of Label Data
Description	About the input data of the algorithm		

Crypto_JobType

Type	Parameter	Description
uint32	cryptoKeyId	RSA Key Number

- The cryptoKey must have a valid value set in the following Key Element before the service can be executed.

key element Name	key element ID	Comments
CRYPTO_KE_CIPHER_KEY	1	RSA private key value (512-byte = N D)

	Document No.	AUTOSAR HaeModule Crypto Driver User Manual	Revision Date	'24. 04
	-		Revision No.	1.03b

6.1.6. CRYPTO_AEADENCRYPT

6.1.6.1. AES GCM

Crypto_PrimitiveInfoType

Service	Algorithm Family	Algorithm Secondary Family	Algorithm Mode
CRYPTO_AEADENCRYPT	CRYPTO_ALGOFAM_AES	CRYPTO_ALGOFAM_NOT_SET	CRYPTO_ALGOMODE_GCM

Crypto_JobPrimitiveInputOutputType

Type	Parameter	Description
const uint8*	inputPtr	Plaintext Data Pointer
uint32	inputLength	Plaintext Data Byte Size
const uint8*	secondaryInputPtr	AAD Data Pointer
uint32	secondaryInputLength	Byte size of AAD data
uint8*	outputPtr	Encrypted data pointer
uint32*	outputLengthPtr	Encrypted data length pointer
uint8*	secondaryOutputPtr	Validation of Ciphertext Tag Data Pointer
uint32*	secondaryOutputLengthPtr	Verification of Ciphertext Tag Data Byte Size

- **HaeCryptoLibPrimitives Caution Guide**
 - When using CRYPTO_OPERATIONMODE_SINGLECALL, g_cryptoLib_gcmLastUpdateCall = 1 is required
 - In other modes, when entering a data block, g_cryptoLib_gcmLastUpdateCall = 0 and inputPtr can only be entered in multiples of 16 bytes. When entering the last block of data, after setting g_cryptoLib_gcmLastUpdateCall = 1, it can be anything other than 16 bytes in length. For additional information, please refer to the CryptoLib manual.
- **HaeHsmPrimitives Caution Guide**
 - Due to the limitation of the functionality of the HSM security module, the input data to be computed and the length of the AAD are only up to 128 bytes.
 - In CRYPTO_OPERATIONMODE_UPDATE mode, all parameters such as input data , AAD, tag, and output data storage pointer must be set according to the normal operation of the HSM Driver API.
 - Therefore, it uses CRYPTO_OPERATIONMODE_SINGLECALL mode of operation.

Crypto_AlgorithmInfoType

Type	Parameter	Description
uint32	keyLength	The key length in bytes to be used with that algorithm Applicable AES Key Byte Length <ul style="list-style-type: none"> • AES-128 : 16 • AES-192: 24 (not allowed by HaeHsmPrimitives) • AES-256 : 32

	Document No.	AUTOSAR HaeModule Crypto Driver User Manual	Revision Date	'24. 04
	-		Revision No.	1.03b

- The application algorithm is determined by the setting value.

Crypto_JobType

Type	Parameter	Description
uint32	cryptoKeyId	AES GCM Key Number

- The cryptoKey must have a valid value set in the following Key Element before the service can be executed.

key element Name	key element ID	Comments
CRYPTO_KE_CIPHER_KEY	1	AES GCM Key Values
CRYPTO_KE_CIPHER_IV	5	AES GCM IV values

	Document No.	AUTOSAR HaeModule Crypto Driver User Manual	Revision Date	'24. 04
	-		Revision No.	1.03b

6.1.6.2. CHACHA20-POLY1305

Crypto_PrimitiveInfoType < [HaeCryptoLibPrimitives](#) >

Service	Algorithm Family	Algorithm Secondary Family	Algorithm Mode
CRYPTO_AEADENCRYPT	CRYPTO_ALGOFAM_CHACHA	CRYPTO_ALGOFAM_POLY1305	CRYPTO_ALGOMODE_20ROUNDS

Crypto_JobPrimitiveInputOutputType

Type	Parameter	Description
const uint8*	inputPtr	Plaintext Data Pointer
uint32	inputLength	Plaintext Data Byte Size
const uint8*	secondaryInputPtr	AAD Data Pointer
uint32	secondaryInputLength	Byte size of AAD data
uint8*	outputPtr	Encrypted data pointer
uint32*	outputLengthPtr	Encrypted data length pointer
uint8*	secondaryOutputPtr	Validation of Ciphertext Tag Data Pointer
uint32*	secondaryOutputLengthPtr	Verification of Ciphertext Tag Data Byte Size

Crypto_JobType

Type	Parameter	Description
uint32	cryptoKeyId	CHACHA20-POLY1305 Key Number

- The cryptoKey must have a valid value set in the following Key Element before the service can be executed.

key element Name	key element ID	Comments
CRYPTO_KE_CIPHER_KEY	1	CHACHA20-POLY1305 key value (32-byte)
CRYPTO_KE_CIPHER_IV	5	CHACHA20-POLY1305 nonce value (12-byte)

	Document No.	AUTOSAR HaeModule Crypto Driver User Manual	Revision Date	'24. 04
	-		Revision No.	1.03b

6.1.7. CRYPTO_AEADDECRYPT

6.1.7.1. AES GCM

Crypto_PrimitiveInfoType

Service	Algorithm Family	Algorithm Secondary Family	Algorithm Mode
CRYPTO_AEADDECRYPT	CRYPTO_ALGOFAM_AES	CRYPTO_ALGOFAM_NOT_SET	CRYPTO_ALGOMODE_GCM

Crypto_JobPrimitiveInputOutputType

Type	Parameter	Description
const uint8*	inputPtr	Data pointer to decrypt
uint32	inputLength	The size of the data bytes to be decrypted.
const uint8*	secondaryInputPtr	AAD Data Pointer
uint32	secondaryInputLength	Byte size of AAD data
const uint8*	tertiaryInputPtr	Validation of Ciphertext Tag Data Pointer
uint32	tertiaryInputLength	Verification of Ciphertext Tag Data Byte Size
uint8*	outputPtr	Decrypted data storage pointer
uint32*	outputLengthPtr	Decrypted data length pointer
Crypto_VerifyResultType*	verifyPtr	Verification Results

- **HaeCryptoLibPrimitives Caution Guide**
 - When using CRYPTO_OPERATIONMODE_SINGLECALL, g_cryptoLib_gcmLastUpdateCall = 1 is required
 - In other modes, when entering a data block, g_cryptoLib_gcmLastUpdateCall = 0 and inputPtr can only be entered in multiples of 16 bytes. When entering the last block of data, after setting g_cryptoLib_gcmLastUpdateCall = 1, it can be anything other than 16 bytes in length. For additional information, please refer to the CryptoLib manual.
- **HaeHsmPrimitives Caution Guide**
 - Due to the limitation of the functionality of the HSM security module, the input data to be computed and the length of the AAD are only up to 128 bytes.
 - In CRYPTO_OPERATIONMODE_UPDATE mode, all parameters such as input data, AAD, tag, and output data storage pointer must be set according to the normal operation of the HSM Driver API.
 - Therefore, it uses CRYPTO_OPERATIONMODE_SINGLECALL mode of operation.

Crypto_AlgorithmInfoType

Type	Parameter	Description
uint32	keyLength	The key length in bytes to be used with that algorithm Applicable AES Key Byte Length <ul style="list-style-type: none"> • AES-128 : 16 • AES-192: 24 (not allowed by HaeHsmPrimitives)

	Document No.	AUTOSAR HaeModule Crypto Driver User Manual	Revision Date	'24. 04
	-		Revision No.	1.03b

		• AES-256 : 32
--	--	----------------

- The application algorithm is determined by the setting value.

Crypto_JobType

Type	Parameter	Description
uint32	cryptoKeyId	AES GCM Key Number

- The cryptoKey must have a valid value set in the following Key Element before the service can be executed.

key element Name	key element ID	Comments
CRYPTO_KE_CIPHER_KEY	1	AES GCM Key Values
CRYPTO_KE_CIPHER_IV	5	AES GCM IV values

	Document No.	AUTOSAR HaeModule Crypto Driver User Manual	Revision Date	'24. 04
	-		Revision No.	1.03b

6.1.7.2. CHACHA20-POLY1305

Crypto_PrimitiveInfoType < [HaeCryptoLibPrimitives](#) >

Service	Algorithm Family	Algorithm Secondary Family	Algorithm Mode
CRYPTO_AEADDECRYPT	CRYPTO_ALGOFAM_CHACHA	CRYPTO_ALGOFAM_POLY1305	CRYPTO_ALGOMODE_20ROUNDS

Crypto_JobPrimitiveInputOutputType

Type	Parameter	Description
const uint8*	inputPtr	Data pointer to decrypt
uint32	inputLength	The size of the data bytes to be decrypted.
const uint8*	secondaryInputPtr	AAD Data Pointer
uint32	secondaryInputLength	Byte size of AAD data
const uint8*	tertiaryInputPtr	Validation of Ciphertext Tag Data Pointer
uint32	tertiaryInputLength	Verification of Ciphertext Tag Data Byte Size
uint8*	outputPtr	Decrypted data storage pointer
uint32*	outputLengthPtr	Decrypted data length pointer
Crypto_VerifyResultType*	verifyPtr	Verification Results

Crypto_JobType

Type	Parameter	Description
uint32	cryptoKeyId	CHACHA20-POLY1305 Key Number

- The cryptoKey must have a valid value set in the following Key Element before the service can be executed.

key element Name	key element ID	Comments
CRYPTO_KE_CIPHER_KEY	1	CHACHA20-POLY1305 key value (32-byte)
CRYPTO_KE_CIPHER_IV	5	CHACHA20-POLY1305 nonce value (12-byte)

	Document No.	AUTOSAR HaeModule Crypto Driver User Manual	Revision Date	'24. 04
	-		Revision No.	1.03b

6.1.8. CRYPTO_SIGNATUREGENERATE

6.1.8.1. RSASSA-PKCS1_v1_5

Crypto_PrimitiveInfoType < HaeHsmPrimitives >

Service	Algorithm Family	Algorithm Secondary Family	Algorithm Mode
CRYPTO_SIGNATUREGENERATE	CRYPTO_ALGOFAM_RSA	CRYPTO_ALGOFAM_SHA2_256	CRYPTO_ALGOMODE_RSASSA_PKCS1_v1_5

Crypto_PrimitiveInfoType < HaeCryptoLibPrimitives >

Service	Algorithm Family	Algorithm Secondary Family	Algorithm Mode
CRYPTO_SIGNATUREGENERATE	CRYPTO_ALGOFAM_RSA	CRYPTO_ALGOFAM_SHA1	CRYPTO_ALGOMODE_RSASSA_PKCS1_v1_5
CRYPTO_SIGNATUREGENERATE	CRYPTO_ALGOFAM_RSA	CRYPTO_ALGOFAM_SHA2_224	CRYPTO_ALGOMODE_RSASSA_PKCS1_v1_5
CRYPTO_SIGNATUREGENERATE	CRYPTO_ALGOFAM_RSA	CRYPTO_ALGOFAM_SHA2_256	CRYPTO_ALGOMODE_RSASSA_PKCS1_v1_5
CRYPTO_SIGNATUREGENERATE	CRYPTO_ALGOFAM_RSA	CRYPTO_ALGOFAM_SHA2_384	CRYPTO_ALGOMODE_RSASSA_PKCS1_v1_5
CRYPTO_SIGNATUREGENERATE	CRYPTO_ALGOFAM_RSA	CRYPTO_ALGOFAM_SHA2_512	CRYPTO_ALGOMODE_RSASSA_PKCS1_v1_5

Crypto_JobPrimitiveInputOutputType

Type	Parameter	Description
const uint8*	inputPtr	Plaintext Data Pointer
uint32	inputLength	inputPtr byte size
uint8*	outputPtr	Generated signature value of 256 bytes
uint32*	outputLengthPtr	outputPtr byte size

Crypto_JobType

Type	Parameter	Description
uint32	cryptoKeyId	RSA Key Number

- The cryptoKey must have a valid value set in the following Key Element before the service can be executed.

key element Name	key element ID	Comments
CRYPTO_KE_SIGNATURE_KEY	1	RSA private key value (512-byte = N D)

	Document No.	AUTOSAR HaeModule Crypto Driver User Manual	Revision Date	'24. 04
	-		Revision No.	1.03b

6.1.8.2. RSASSA-PSS

Crypto_PrimitiveInfoType < HaeHsmPrimitives >

Service	Algorithm Family	Algorithm Secondary Family	Algorithm Mode
CRYPTO_SIGNATUREGENERATE	CRYPTO_ALGOFAM_RSA	CRYPTO_ALGOFAM_SHA2_256	CRYPTO_ALGOMODE_RSASSA_PSS

Crypto_PrimitiveInfoType < HaeCryptoLibPrimitives >

Service	Algorithm Family	Algorithm Secondary Family	Algorithm Mode
CRYPTO_SIGNATUREGENERATE	CRYPTO_ALGOFAM_RSA	CRYPTO_ALGOFAM_SHA1	CRYPTO_ALGOMODE_RSASSA_PSS
CRYPTO_SIGNATUREGENERATE	CRYPTO_ALGOFAM_RSA	CRYPTO_ALGOFAM_SHA2_224	CRYPTO_ALGOMODE_RSASSA_PSS
CRYPTO_SIGNATUREGENERATE	CRYPTO_ALGOFAM_RSA	CRYPTO_ALGOFAM_SHA2_256	CRYPTO_ALGOMODE_RSASSA_PSS
CRYPTO_SIGNATUREGENERATE	CRYPTO_ALGOFAM_RSA	CRYPTO_ALGOFAM_SHA2_384	CRYPTO_ALGOMODE_RSASSA_PSS

Crypto_JobPrimitiveInputOutputType

Type	Parameter	Description
const uint8*	inputPtr	Plaintext data pointer (for HaeHsmPrimitive, enter as HSM_RSA_PKCS1_PSS_SIGNGENERATE_INPUT_t)
uint32	inputLength	inputPtr byte size
uint8*	outputPtr	Generated signature value of 256 bytes
uint32*	outputLengthPtr	outputPtr byte size

Crypto_JobType

Type	Parameter	Description
uint32	cryptoKeyId	RSA Key Number

- The cryptoKey must have a valid value set in the following Key Element before the service can be executed.

key element Name	key element ID	Comments
CRYPTO_KE_SIGNATURE_KEY	1	RSA private key value (512-byte = N D)
CRYPTO_KE_SIGNATURE_SALT	1013	RSASSA-PSS salt value (HaeCryptoLibPrimitive only)

	Document No.	AUTOSAR HaeModule Crypto Driver User Manual	Revision Date	'24. 04
	-		Revision No.	1.03b

6.1.8.3. ECDSA

Crypto_PrimitiveInfoType < **HaeHsmPrimitives** >

Service	Algorithm Family	Algorithm Secondary Family	Algorithm Mode
CRYPTO_SIGNATUREGENERATE	CRYPTO_ALGOFAM_ECCNIST	CRYPTO_ALGOFAM_SHA2_256	CRYPTO_ALGOMODE_NOT_SET
CRYPTO_SIGNATUREGENERATE	CRYPTO_ALGOFAM_ECCNIST	CRYPTO_ALGOFAM_SHA2_512	CRYPTO_ALGOMODE_NOT_SET

- SHA256 only offers P256R1 with a curve type, and SHA512 only gives P521R1 curve type.

Crypto_PrimitiveInfoType < **HaeCryptoLibPrimitives** >

Service	Algorithm Family	Algorithm Secondary Family	Algorithm Mode
CRYPTO_SIGNATUREGENERATE	CRYPTO_ALGOFAM_ECCNIST	CRYPTO_ALGOFAM_SHA1	CRYPTO_ALGOMODE_NOT_SET
CRYPTO_SIGNATUREGENERATE	CRYPTO_ALGOFAM_ECCNIST	CRYPTO_ALGOFAM_SHA2_224	CRYPTO_ALGOMODE_NOT_SET
CRYPTO_SIGNATUREGENERATE	CRYPTO_ALGOFAM_ECCNIST	CRYPTO_ALGOFAM_SHA2_256	CRYPTO_ALGOMODE_NOT_SET
CRYPTO_SIGNATUREGENERATE	CRYPTO_ALGOFAM_ECCNIST	CRYPTO_ALGOFAM_SHA2_384	CRYPTO_ALGOMODE_NOT_SET
CRYPTO_SIGNATUREGENERATE	CRYPTO_ALGOFAM_ECCNIST	CRYPTO_ALGOFAM_SHA2_512	CRYPTO_ALGOMODE_NOT_SET

Crypto_JobPrimitiveInputOutputType

Type	Parameter	Description
const uint8*	inputPtr	Plaintext Data Pointer
uint32	inputLength	inputPtr byte size
uint8*	outputPtr	Signature values that are generated
uint32*	outputLengthPtr	outputPtr byte size

- The size of the signature value generated depends on the Curve Type.
HAEMODULE_CRYPTOCURVETYPE_P160R1 : 2 * 20bytes
HAEMODULE_CRYPTOCURVETYPE_P224R1 : 2 * 28bytes
HAEMODULE_CRYPTOCURVETYPE_P256R1 : 2 * 32bytes
HAEMODULE_CRYPTOCURVETYPE_P521R1 : 2 * 66bytes

Crypto_JobType

Type	Parameter	Description
uint32	cryptoKeyId	ECDSA Key Number

- The cryptoKey must have a valid value set in the following Key Element before the service can be executed.

key element Name	key element ID	Comments
CRYPTO_KE_SIGNATURE_KEY	1	ECC Private Key Value
CRYPTO_KE_SIGNATURE_SALT	1013	ECDSA salt value (optional, for HaeHsmPrimitives)
CRYPTO_KE_SIGNATURE_CURVETYPE	29	Vendor ECC Curve Type ID (4-byte, only the algorithm 'Curve Key Length' = Hash Output Length' for each curve type is provided)

- The defined Curve Type ID can be found in Crypto_76_HaeModule_UserTypes.h.

	Document No.	AUTOSAR HaeModule Crypto Driver User Manual	Revision Date	'24. 04
	-		Revision No.	1.03b

- 1. HAEMODULE_CRYPTOCURVETYPE_P160R1 = 0x01
- 2. HAEMODULE_CRYPTOCURVETYPE_P224R1 = 0x02
- 3. HAEMODULE_CRYPTOCURVETYPE_P256R1 = 0x03
- 4. HAEMODULE_CRYPTOCURVETYPE_P521R1 = 0x05

	Document No.	AUTOSAR HaeModule Crypto Driver User Manual	Revision Date	'24. 04
	-		Revision No.	1.03b

6.1.8.4. EDDSA ED448

Crypto_PrimitiveInfoType

Service	Algorithm Family	Algorithm Secondary Family	Algorithm Mode
CRYPTO_SIGNATUREGENERATE	CRYPTO_ALGOFAM_ECCNIST	CRYPTO_ALGOFAM_SHAKE256	CRYPTO_ALGOMODE_NOT_SET

Crypto_JobPrimitiveInputOutputType

Type	Parameter	Description
const uint8*	inputPtr	HAEMODULE_EDDSA_ED448_INTPUT_t Pointer
uint32	inputLength	inputPtr byte size
uint8*	outputPtr	114 bytes signature value generated
uint32*	outputLengthPtr	outputPtr byte size

- Update operation mode can only be called once.

HAEMODULE_EDDSA_ED448_INTPUT_t

Name	HAEMODULE_EDDSA_ED448_INTPUT_t		
Type	Structure		
Element	uint8*	message	Message to sign
	uint32	messageLength	The length of the message to be signed
	uint8*	context	Context required for signing (less than 256 bytes)
	uint32	contextLength	Length of Context
Description	About the Input Data of the ECC EDDSA ED448 Algorithm		

- HaeHsmPrimitives Caution Guide**
- Due to the functionality limitations of the HSM security module, the input data message and context to be computed are only supported by a maximum of 64 bytes.

Crypto_JobType

Type	Parameter	Description
uint32	cryptoKeyId	ECC Key Number

- The cryptoKey must have a valid value set in the following Key Element before the service can be executed.

key element Name	key element ID	Comments
CRYPTO_KE_SIGNATURE_KEY	1	ED448 Private Key Value

- Currently, EDDSA only supports ED448, so there is no need to set a separate curve type.

	Document No.	AUTOSAR HaeModule Crypto Driver User Manual	Revision Date	'24. 04
	-		Revision No.	1.03b

6.1.9. CRYPTO_SIGNATUREVERIFY

6.1.9.1. RSASSA-PKCS1_v1_5

Crypto_PrimitiveInfoType < HaeHsmPrimitives >

Service	Algorithm Family	Algorithm Secondary Family	Algorithm Mode
CRYPTO_SIGNATUREVERIFY	CRYPTO_ALGOFAM_RSA	CRYPTO_ALGOFAM_SHA1	CRYPTO_ALGOMODE_RSASSA_PKCS1_v1_5
CRYPTO_SIGNATUREVERIFY	CRYPTO_ALGOFAM_RSA	CRYPTO_ALGOFAM_SHA2_256	CRYPTO_ALGOMODE_RSASSA_PKCS1_v1_5

Crypto_PrimitiveInfoType < HaeCryptoLibPrimitives >

Service	Algorithm Family	Algorithm Secondary Family	Algorithm Mode
CRYPTO_SIGNATUREVERIFY	CRYPTO_ALGOFAM_RSA	CRYPTO_ALGOFAM_SHA1	CRYPTO_ALGOMODE_RSASSA_PKCS1_v1_5
CRYPTO_SIGNATUREVERIF	CRYPTO_ALGOFAM_RSA	CRYPTO_ALGOFAM_SHA2_224	CRYPTO_ALGOMODE_RSASSA_PKCS1_v1_5
CRYPTO_SIGNATUREVERIFY	CRYPTO_ALGOFAM_RSA	CRYPTO_ALGOFAM_SHA2_256	CRYPTO_ALGOMODE_RSASSA_PKCS1_v1_5
CRYPTO_SIGNATUREVERIFY	CRYPTO_ALGOFAM_RSA	CRYPTO_ALGOFAM_SHA2_384	CRYPTO_ALGOMODE_RSASSA_PKCS1_v1_5
CRYPTO_SIGNATUREVERIFY	CRYPTO_ALGOFAM_RSA	CRYPTO_ALGOFAM_SHA2_512	CRYPTO_ALGOMODE_RSASSA_PKCS1_v1_5

Crypto_JobPrimitiveInputOutputType

Type	Parameter	Description
const uint8*	inputPtr	Plaintext Data Pointer
uint32	inputLength	inputPtr byte size
const uint8*	secondaryInputPtr	256 bytes of signature value to be verified
uint32	secondaryInputLength	secondaryInputPtr byte size
Crypto_VerifyResultType*	verifyPtr	Validation Result Values


- If the highest byte of the signature value is 0x00, input up to +1 byte is allowed.

Crypto_JobType

Type	Parameter	Description
uint32	cryptoKeyId	RSA Key Number

- The cryptoKey must have a valid value set in the following Key Element before the service can be executed.

key element Name	key element ID	Comments
CRYPTO_KE_SIGNATURE_KEY	1	RSA public key value (260-byte = N E (4-byte))

	Document No.	AUTOSAR HaeModule Crypto Driver User Manual	Revision Date	'24. 04
	-		Revision No.	1.03b

6.1.9.2. RSASSA-PSS

Crypto_PrimitiveInfoType < HaeHsmPrimitives >

Service	Algorithm Family	Algorithm Secondary Family	Algorithm Mode
CRYPTO_SIGNATUREVERIFY	CRYPTO_ALGOFAM_RSA	CRYPTO_ALGOFAM_SHA2_256	CRYPTO_ALGOMODE_RSASSA_PSS

Crypto_PrimitiveInfoType < HaeCryptoLibPrimitives >

Service	Algorithm Family	Algorithm Secondary Family	Algorithm Mode
CRYPTO_SIGNATUREVERIFY	CRYPTO_ALGOFAM_RSA	CRYPTO_ALGOFAM_SHA1	CRYPTO_ALGOMODE_RSASSA_PSS
CRYPTO_SIGNATUREVERIFY	CRYPTO_ALGOFAM_RSA	CRYPTO_ALGOFAM_SHA2_224	CRYPTO_ALGOMODE_RSASSA_PSS
CRYPTO_SIGNATUREVERIFY	CRYPTO_ALGOFAM_RSA	CRYPTO_ALGOFAM_SHA2_256	CRYPTO_ALGOMODE_RSASSA_PSS
CRYPTO_SIGNATUREVERIFY	CRYPTO_ALGOFAM_RSA	CRYPTO_ALGOFAM_SHA2_384	CRYPTO_ALGOMODE_RSASSA_PSS

Crypto_JobPrimitiveInputOutputType

Type	Parameter	Description
const uint8*	inputPtr	Plaintext Data Pointer (For HaeHsmPrimitive, enter as HSM_RSA_PKCS1_PSS_SIGNVERIFY_INTPUT_t)
uint32	inputLength	inputPtr byte size
const uint8*	secondaryInputPtr	256 bytes of signature value to be verified
uint32	secondaryInputLength	secondaryInputPtr byte size
Crypto_VerifyResultType*	verifyPtr	Validation Result Values

- If the highest byte of the signature value is 0x00, input up to +1 byte is allowed.

Crypto_JobType

Type	Parameter	Description
uint32	cryptoKeyId	RSA Key Number

- The cryptoKey must have a valid value set in the following Key Element before the service can be executed.

key element Name	key element ID	Comments
CRYPTO_KE_SIGNATURE_KEY	1	RSA public key value (260-byte = N E (4-byte))
CRYPTO_KE_SIGNATURE_SALT	1013	Byte length of RSASSA-PSS salt values (in 4-byte format) (HaeCryptoLibPrimitives only)

	Document No.	AUTOSAR HaeModule Crypto Driver User Manual	Revision Date	'24. 04
	-		Revision No.	1.03b

6.1.9.3. ECDSA

Crypto_PrimitiveInfoType < HaeHsmPrimitives >

Service	Algorithm Family	Algorithm Secondary Family	Algorithm Mode
CRYPTO_SIGNATUREVERIFY	CRYPTO_ALGOFAM_ECCNIST	CRYPTO_ALGOFAM_SHA2_256	CRYPTO_ALGOMODE_NOT_SET
CRYPTO_SIGNATUREVERIFY	CRYPTO_ALGOFAM_ECCNIST	CRYPTO_ALGOFAM_SHA2_512	CRYPTO_ALGOMODE_NOT_SET

- SHA256 only gives P256R1 a curve type, and SHA512 gives only P521R1 curve type.

Crypto_PrimitiveInfoType < HaeCryptoLibPrimitives >

Service	Algorithm Family	Algorithm Secondary Family	Algorithm Mode
CRYPTO_SIGNATUREVERIFY	CRYPTO_ALGOFAM_ECCNIST	CRYPTO_ALGOFAM_SHA1	CRYPTO_ALGOMODE_NOT_SET
CRYPTO_SIGNATUREVERIFY	CRYPTO_ALGOFAM_ECCNIST	CRYPTO_ALGOFAM_SHA2_224	CRYPTO_ALGOMODE_NOT_SET
CRYPTO_SIGNATUREVERIFY	CRYPTO_ALGOFAM_ECCNIST	CRYPTO_ALGOFAM_SHA2_256	CRYPTO_ALGOMODE_NOT_SET
CRYPTO_SIGNATUREVERIFY	CRYPTO_ALGOFAM_ECCNIST	CRYPTO_ALGOFAM_SHA2_384	CRYPTO_ALGOMODE_NOT_SET
CRYPTO_SIGNATUREVERIFY	CRYPTO_ALGOFAM_ECCNIST	CRYPTO_ALGOFAM_SHA2_512	CRYPTO_ALGOMODE_NOT_SET

Crypto_JobPrimitiveInputOutputType

Type	Parameter	Description
const uint8*	inputPtr	Plaintext Data Pointer
uint32	inputLength	inputPtr byte size
const uint8*	secondaryInputPtr	Signature values to be validated
uint32	secondaryInputLength	secondaryInputPtr byte size
Crypto_VerifyResultType*	verifyPtr	Validation Result Values

- If the byte higher r and s of the signature value is 0x00, then +1byte input is allowed, up to +2byte.
- The size of the signature value to be verified depends on the Curve Type.
HAEMODULE_CRYPTOCURVETYPE_P160R1 : 2 * 20bytes
HAEMODULE_CRYPTOCURVETYPE_P224R1 : 2 * 28bytes
HAEMODULE_CRYPTOCURVETYPE_P256R1 : 2 * 32bytes
HAEMODULE_CRYPTOCURVETYPE_P521R1 : 2 * 66bytes

Crypto_JobType

Type	Parameter	Description
uint32	cryptoKeyld	ECC Key Number

- The cryptoKey must have a valid value set in the following Key Element before the service can be executed.

key element Name	key element ID	Comments
CRYPTO_KE_SIGNATURE_KEY	1	ECC Public Key Value
CRYPTO_KE_SIGNATURE_SALT	1013	ECDSA salt value (optional, for HaeHsmPrimitives)
CRYPTO_KE_SIGNATURE_CURVETYPE	29	Vendor ECC Curve Type ID (4-byte, only provides algorithms with 'Curve Key Length >= Hash Output Length' for each curve type)

	Document No.	AUTOSAR HaeModule Crypto Driver User Manual	Revision Date	'24. 04
	-		Revision No.	1.03b

- The defined Curve Type ID can be found in Crypto_76_HaeModule_UserTypes.h.
 1. HAEMODULE_CRYPTOCURVETYPE_P160R1 = 0x01
 2. HAEMODULE_CRYPTOCURVETYPE_P224R1 = 0x02
 3. HAEMODULE_CRYPTOCURVETYPE_P256R1 = 0x03
 4. HAEMODULE_CRYPTOCURVETYPE_P521R1 = 0x05

	Document No.	AUTOSAR HaeModule Crypto Driver User Manual	Revision Date	'24. 04
	-		Revision No.	1.03b

6.1.9.4. EDDSA ED448

Crypto_PrimitiveInfoType

Service	Algorithm Family	Algorithm Secondary Family	Algorithm Mode
CRYPTO_SIGNATUREVERIFY	CRYPTO_ALGOFAM_ECCNIST	CRYPTO_ALGOFAM_SHAKE256	CRYPTO_ALGOMODE_NOT_SET

Crypto_JobPrimitiveInputOutputType

Type	Parameter	Description
const uint8*	inputPtr	HAEMODULE_EDDSA_ED448_INTPUT_t Pointer
uint32	inputLength	inputPtr byte size
const uint8*	secondaryInputPtr	114 bytes signature value to validate
uint32	secondaryInputLength	outputPtr byte size
Crypto_VerifyResultType*	verifyPtr	Verification Results

- If the upper byte of r and s of the signature value is 0x00, +1byte input is allowed, up to +2byte.
- HaeHsmPrimitives Caution Guide**
 - For ECC_EDDSA_ED448_SIGN_t, please refer to the HSM Module User Manual.

HAEMODULE_EDDSA_ED448_INTPUT_t

Name	HAEMODULE_EDDSA_ED448_INTPUT_t		
Type	Structure		
Element	uint8*	message	Message to sign
	uint32	messageLength	The length of the message to be signed
	uint8*	context	Context required for signing (less than 256 bytes)
	uint32	contextLength	Length of Context
Description	About the Input Data of the ECC EDDSA ED448 Algorithm		

- HaeHsmPrimitives Caution Guide**
- Due to the functionality limitations of the HSM security module, the input data message and context to be computed are only supported by a maximum of 64 bytes.

Crypto_JobType

Type	Parameter	Description
uint32	cryptoKeyId	ECC Key Number

- The cryptoKey must have a valid value set in the following Key Element before the service can be executed.

key element Name	key element ID	Comments
CRYPTO_KE_SIGNATURE_KEY	1	ED448 Public Key Value

- Currently, EDDSA only supports ED448, so there is no need to set a separate curve type.

	Document No.	AUTOSAR HaeModule Crypto Driver User Manual	Revision Date	'24. 04
	-		Revision No.	1.03b

	Document No.	AUTOSAR HaeModule Crypto Driver User Manual	Revision Date	'24. 04
	-		Revision No.	1.03b

6.1.10. CRYPTO_RANDOMGENERATE

6.1.10.1. DRBG

Caution:

DRBG requires a seed to be entered.

Therefore, you must use the RandomSeed service before using RandomGenerate, and you must also use the RandomSeed service when returning an error while using RandomGenerate.

- CSM R4.4 CsmJobKeyDeriveAlgorithmFamily not supported
 - After setting up the CRYPTO_ALGOFAM_CUSTOM, enter the value in the AlgorithmFamilyCustom
- CSM R4.4 CsmJobKeyDeriveAlgorithmSecondaryFamily not supported
 - After setting the CRYPTO_ALGOFAM_CUSTOM, enter a value in the AlgorithmSecondaryFamilyCustom

Crypto_PrimitiveInfoType < [HaeCryptoLibPrimitives](#) >

Service	Algorithm Family	Algorithm Secondary Family	Algorithm Mode
CRYPTO_RANDOMGENERATE	CRYPTO_ALGOFAM_DRBG	CRYPTO_ALGOFAM_SHA2_256	CRYPTO_ALGOMODE_NOT_SET

Crypto_JobPrimitiveInputOutputType

Type	Parameter	Description
uint8*	outputPtr	Random Number Data Pointer
uint32*	outputLengthPtr	Random Number Data Length Pointer If you enter a value greater than 0 or 1024, E_PARAM_VALUE occurrence

6.1.10.2. PRNG

Crypto_PrimitiveInfoType < [HaeHsmPrimitives](#) >

Service	Algorithm Family	Algorithm Secondary Family	Algorithm Mode
CRYPTO_RANDOMGENERATE	CRYPTO_ALGOFAM_RNG	CRYPTO_ALGOFAM_NOT_SET	CRYPTO_ALGOMODE_NOT_SET

Crypto_JobPrimitiveInputOutputType

Type	Parameter	Description
uint8*	outputPtr	Random Number Data Pointer
uint32*	outputLengthPtr	Random Number Data Length Pointer

	Document No.	AUTOSAR HaeModule Crypto Driver User Manual	Revision Date	'24. 04
	-		Revision No.	1.03b

		If you enter a value greater than 0 or 1024, E_PARAM_VALUE occurrence
--	--	--

6.1.10.3. TRNG

Crypto_PrimitiveInfoType < [HaeHsmPrimitives](#) >

Service	Algorithm Family	Algorithm Secondary Family	Algorithm Mode
CRYPTO_RANDOMGENERATE	CRYPTO_ALGOFAM_RNG	CRYPTO_ALGOFAM_NOT_SET	CRYPTO_ALGOMODE_CUSTOM

Crypto_JobPrimitiveInputOutputType

Type	Parameter	Description
uint8*	outputPtr	Random Number Data Pointer
uint32*	outputLengthPtr	Random Number Data Length Pointer If you enter a value greater than 0 or 1024, E_PARAM_VALUE occurrence

	Document No.	AUTOSAR HaeModule Crypto Driver User Manual	Revision Date	'24. 04
	-		Revision No.	1.03b

6.1.11. CRYPTO_RANDOMSEED

6.1.11.1. DRBG

- CSM R4.4 CsmJobKeyDeriveAlgorithmFamily not supported
 - After setting up the CRYPTO_ALGOFAM_CUSTOM, enter the value in the AlgorithmFamilyCustom
- CSM R4.4 CsmJobKeyDeriveAlgorithmSecondaryFamily not supported
 - After setting the CRYPTO_ALGOFAM_CUSTOM, enter a value in the AlgorithmSecondaryFamilyCustom

Crypto_PrimitiveInfoType < [HaeCryptoLibPrimitives](#) >

Service	Algorithm Family	Algorithm Secondary Family	Algorithm Mode
CRYPTO_RANDOMGENERATE	CRYPTO_ALGOFAM_DRBG	CRYPTO_ALGOFAM_SHA2_256	CRYPTO_ALGOMODE_NOT_SET

Crypto_JobPrimitiveInputOutputType

Type	Parameter	Description
uint8*	inputPtr	DRBG's Input Seed Data Pointer (The first seed must be at least 48 bytes long.)
uint32	inputLength	inputPtr byte size

Crypto_JobType


Type	Parameter	Description
uint32	cryptoKeyId	DRBG Key Number

- The cryptoKey must have a valid value set in the following Key Element before the service can be executed.

key element Name	key element ID	Comments
CRYPTO_KE_RANDOM_ALGORITHM	4	Algorithm ID of the vendor DRBG (4-byte when using Key Interface)

- The defined DRBG ID can be found in Crypto_76_HaeModule_UserTypes.h.

1. HAEMODULE_CRYPT0_DRBG_SHA256 = 0x01

	Document No.	AUTOSAR HaeModule Crypto Driver User Manual	Revision Date	'24. 04
	-		Revision No.	1.03b

6.1.12. CRYPTO_KEYEXCHANGE_CALCPUBVAL

6.1.12.1. Diffie-Hellman

Crypto_PrimitiveInfoType

Service	Algorithm Family	Algorithm Secondary Family	Algorithm Mode
CRYPTO_KEYEXCHANGE_CALCPUBVAL	CRYPTO_ALGOFAM_DH	CRYPTO_ALGOFAM_NOT_SET	CRYPTO_ALGOMODE_NOT_SET

Crypto_JobPrimitiveInputOutputType

Type	Parameter	Description
uint8*	outputPtr	A storage pointer of the generated public key value
uint32*	outputLengthPtr	outputPtr byte size (256 bytes)

Crypto_JobType

Type	Parameter	Description
uint32	cryptoKeyId	Key for Key Exchange

- The cryptoKey must have a valid value set in the following Key Element before the service can be executed.

key element Name	key element ID	Comments
CRYPTO_KE_KEYEXCHANGE_BASE	8	Modulus P value and Generator G value connected, 512 byte size
CRYPTO_KE_KEYEXCHANGE_PRIVKEY	9	Private key, 256 bytes in size
CRYPTO_KE_KEYEXCHANGE_ALGORITHM	12	Algorithm ID of the vendor's Key Exchange (4-byte when using the Key Interface)

- The defined Key Exchange ID can be found in Crypto_76_HaeModule_UserTypes.h.

- HAEMODULE_CRYPTODH = 0x01
- HAEMODULE_CRYPTODH_HSM = 0x81

	Document No.	AUTOSAR HaeModule Crypto Driver User Manual	Revision Date	'24. 04
	-		Revision No.	1.03b

6.1.12.2. ECDH

Crypto_PrimitiveInfoType < HaeCryptoLibPrimitives >

Service	Algorithm Family	Algorithm Secondary Family	Algorithm Mode
CRYPTO_KEYEXCHANGE_CALCPUBVAL	CRYPTO_ALGOFAM_DH	CRYPTO_ALGOFAM_ECNIST	CRYPTO_ALGOMODE_NOT_SET

Crypto_JobPrimitiveInputOutputType

Type	Parameter	Description
uint8*	outputPtr	A storage pointer of the generated public key value
uint32*	outputLengthPtr	outputPtr byte size

- The size of the public key value to be generated depends on the Curve Type.
HAEMODULE_CRYPTOCURVETYPE_P160R1 : 2 * 20bytes
HAEMODULE_CRYPTOCURVETYPE_P224R1 : 2 * 28bytes
HAEMODULE_CRYPTOCURVETYPE_P256R1 : 2 * 32bytes
HAEMODULE_CRYPTOCURVETYPE_P521R1 : 2 * 66bytes
HAEMODULE_CRYPTOCURVETYPE_CURVE448 : 56bytes

Crypto_JobType

Type	Parameter	Description
uint32	cryptoKeyld	Key for Key Exchange

- The cryptoKey must have a valid value set in the following Key Element before the service can be executed.

key element Name	key element ID	Comments
CRYPTO_KE_KEYEXCHANGE_PRIVKEY	9	ECC Private Key
CRYPTO_KE_KEYEXCHANGE_CURVETYPE	29	Vendor ECC Curve Type ID (4-byte)
CRYPTO_KE_KEYEXCHANGE_ALGORITHM	12	Algorithm ID of the vendor's Key Exchange (4-byte when using the Key Interface)

- The defined Curve Type ID can be found in Crypto_76_HaeModule_UserTypes.h.
 - HAEMODULE_CRYPTOCURVETYPE_P160R1 = 0x01
 - HAEMODULE_CRYPTOCURVETYPE_P224R1 = 0x02
 - HAEMODULE_CRYPTOCURVETYPE_P256R1 = 0x03
 - HAEMODULE_CRYPTOCURVETYPE_P521R1 = 0x05
 - HAEMODULE_CRYPTOCURVETYPE_CURVE448 = 0x06
- The defined Key Exchange ID can be found in Crypto_76_HaeModule_UserTypes.h.
 - HAEMODULE_CRYPTOECDH = 0x02
 - HAEMODULE_CRYPTOECDH_HSM = 0x82

	Document No.	AUTOSAR HaeModule Crypto Driver User Manual	Revision Date	'24. 04
	-		Revision No.	1.03b

	Document No.	AUTOSAR HaeModule Crypto Driver User Manual	Revision Date	'24. 04
	-		Revision No.	1.03b

6.1.13. CRYPTO_KEYEXCHANGE_CALCSECRET

6.1.13.1. Diffie-Hellman

Crypto_PrimitiveInfoType

Service	Algorithm Family	Algorithm Secondary Family	Algorithm Mode
CRYPTO_KEYEXCHANGE_CALCSECRET	CRYPTO_ALGOFAM_DH	CRYPTO_ALGOFAM_NOT_SET	CRYPTO_ALGOMODE_NOT_SET

Crypto_JobPrimitiveInputOutputType

Type	Parameter	Description
const uint8*	inputPtr	The public key structure of the other member with whom you want to share the secret value.
uint32	inputLength	inputPtr byte size

Crypto_JobType

Type	Parameter	Description
uint32	cryptoKeyId	RSA Key Number

- The cryptoKey must have a valid value set in the following Key Element before the service can be executed.

key element Name	key element ID	Comments
CRYPTO_KE_KEYEXCHANGE_BASE	8	Modulus P-value, 256 byte size
CRYPTO_KE_KEYEXCHANGE_PRIVKEY	9	Private key, 256 bytes in size

- The resulting value is stored in the next Element.

key element Name	key element ID	Comments
CYRPTO_KE_KEYEXCHANGE_SHAREDVALUE	1	Secret Number, 256 byte size Calculate the other party's public key and your private key to generate a secret value.
CRYPTO_KE_KEYEXCHANGE_ALGORITHM	12	Algorithm ID of the vendor's Key Exchange (4-byte when using the Key Interface)

- The defined Key Exchange IDs can be found in Crypto_76_HaeModule_UserTypes.h.

- HAEMODULE_CRYPTODH = 0x01
- HAEMODULE_CRYPTODH_HSM = 0x81

	Document No.	AUTOSAR HaeModule Crypto Driver User Manual	Revision Date	'24. 04
	-		Revision No.	1.03b

6.1.13.2. ECDH

It supports multiple algorithms depending on the type of curve used in the algorithm. The settings of the `Crypto_PrimitiveInfoType` are the same, and the algorithm is applied according to the value set in the 'CRYPTO_KE_KEYEXCHANGE_CURVETYPE' of the `cryptoKey`'s Key Element. Therefore, the user should use the API `Crypto_KeySetValid Crypto_KeyElementSet` to set the necessary Key Element including the 'CRYPTO_KE_KEYEXCHANGE_CURVETYPE' before using this algorithm.

Caution:

Depending on whether your HSM security module is supported, some curve types may not be available. Therefore, check the manual of the HSM you are currently using to see if it is supported.

Crypto_PrimitiveInfoType

Service	Algorithm Family	Algorithm Secondary Family	Algorithm Mode
CRYPTO_KEYEXCHANGE_CALCSECRET	CRYPTO_ALGOFAM_DH	CRYPTO_ALGOMODE_ECCNIST	CRYPTO_ALGOMODE_NOT_SET

Crypto_JobPrimitiveInputOutputType

Type	Parameter	Description
const uint8*	inputPtr	The other party's public key (in the case of PXXXXR1 curve, the x-coordinate and y-coordinate connection)
uint32	inputLength	inputPtr byte size

- The size of the public key value depends on the curve type.
HAEMODULE_CRYPTOCURVETYPE_P160R1 : 2 * 20bytes
HAEMODULE_CRYPTOCURVETYPE_P224R1 : 2 * 28bytes
HAEMODULE_CRYPTOCURVETYPE_P256R1 : 2 * 32bytes
HAEMODULE_CRYPTOCURVETYPE_P521R1 : 2 * 66bytes
HAEMODULE_CRYPTOCURVETYPE_CURVE448 : 56bytes

Crypto_JobType

Type	Parameter	Description
uint32	cryptoKeyId	ECC Key Number

- The `cryptoKey` must have a valid value set in the following Key Element before the service can be executed.

key element Name	key element ID	Comments
CRYPTO_KE_KEYEXCHANGE_PRIVKEY	9	ECC Private Key
CRYPTO_KE_KEYEXCHANGE_CURVETYPE	29	Vendor ECC Curve Type ID (4-byte)
CRYPTO_KE_KEYEXCHANGE_ALGORITHM	12	Algorithm ID of the vendor's Key Exchange (4-byte when using the Key Interface)

- The defined Curve Type ID can be found in `Crypto_76_HaeModule_UserTypes.h`.

	Document No.	AUTOSAR HaeModule Crypto Driver User Manual	Revision Date	'24. 04
	-		Revision No.	1.03b

1. HAEMODULE_CRYPTOCURVETYPE_P160R1 = 0x01
 2. HAEMODULE_CRYPTOCURVETYPE_P224R1 = 0x02
 3. HAEMODULE_CRYPTOCURVETYPE_P256R1 = 0x03
 4. HAEMODULE_CRYPTOCURVETYPE_P521R1 = 0x05
 5. HAEMODULE_CRYPTOCURVETYPE_CURVE448 = 0x06
- The defined Key Exchange ID can be found in Crypto_76_HaeModule_UserTypes.h.
 1. HAEMODULE_CRYPTOECDH = 0x02
 2. HAEMODULE_CRYPTOECDH_HSM = 0x82
 - The resulting value is stored in the next Element.

key element Name	key element ID	Comments
CYRPTO_KE_KEYEXCHANGE_SHAREDVALUE	1	Secret Number. Calculate the other party's public key and your private key to generate a secret value.

- The size of the generated Secure Number value depends on the Curve Type.

HAEMODULE_CRYPTOCURVETYPE_P160R1	:	20bytes	HAEMODULE_CRYPTOCURVETYPE_P224R1	:	28bytes
HAEMODULE_CRYPTOCURVETYPE_P256R1	:	32bytes			
HAEMODULE_CRYPTOCURVETYPE_P521R1	:	66bytes	HAEMODULE_CRYPTOCURVETYPE_CURVE448	:	56bytes

	Document No.	AUTOSAR HaeModule Crypto Driver User Manual	Revision Date	'24. 04
	-		Revision No.	1.03b

6.1.14. CRYPTO_KEYDERIVE

6.1.14.1. PBKDF2

- CSM R4.4 CsmJobKeyDeriveAlgorithmSecondaryFamily not supported (R22-11 supported)

- After setting the CRYPTO_ALGOFAM_CUSTOM, enter a value in the AlgorithmSecondaryFamilyCustom

Crypto_PrimitiveInfoType

Service	Algorithm Family	Algorithm Secondary Family	Algorithm Mode
CRYPTO_KEYDERIVE	CRYPTO_ALGOFAM_PBKDF2	CRYPTO_ALGOFAM_SHA2_256	CRYPTO_ALGOMODE_HMAC

Crypto_JobType

Type	Parameter	Description
uint32	cryptoKeyld	Holds the identifier of the key which is used for key derivation.
uint32	targetCryptoKeyld	Holds the identifier of the key which is used to store the derived key.

- The cryptoKey must have a valid value set in the following Key Element before the service can be executed.
- In this case, the cryptoKey and the targetCryptoKey must be set to different keys.

key element Name	key element ID	Comments
CRYPTO_KE_KEYDERIVATION_PASSWORD	1	PBKDF2 Password Value
CRYPTO_KE_KEYDERIVATION_HSMKEYINDEX	1001	PBKDF2 HSM Key index value (4-byte when using Key Interface)
CRYPTO_KE_KEYDERIVATION_SALT	13	PBKDF2 salt value
CRYPTO_KE_KEYDERIVATION_ITERATIONS	14	PBKDF2 iteration value
CRYPTO_KE_KEYDERIVATION_ALGORITHM	15	Algorithm ID of vendor key derivation (4-byte when using Key Interface)


- The defined Key Derivation ID can be found in Crypto_76_HaeModule_UserTypes.h.

0. HAEMODULE_CRYPTOPBKDF2_HMAC_SHA256 = 0x01

1. HAEMODULE_CRYPTOPBKDF2_HMAC_SHA256_HSM = 0x81

- **HaeHsmPrimitives Specialized Feature Guide**

- The password required as input by PBKDF2 can be received as input from the user, or it can be replaced with a key value that is pre-stored in the security area of the HSM Framework. (For details, please refer to the HSM_Framework_UserManual_Crypto_Service manual)
- If you want to replace password with an internal HSM key, you must have a valid HSM **Key Index** set in the KeyElementValue of the CRYPTO_KE_KEYDERIVATION_HSMKEYINDEX. (**HSM Key Type/HSM Key Index should not be set.**)
- If you want to change the HSM Key Index during runtime to perform the job, you can use the

	Document No.	AUTOSAR HaeModule Crypto Driver User Manual	Revision Date	'24. 04
	-		Revision No.	1.03b

Crypto_KeyElementSet and Crypto_KeySetValid API to change the HSM Key Index to the desired HSM Key Index for the CRYPTO_KE_KEYDERIVATION_HSMKEYINDEX. (Example of use.
KeyElementSet(CRYPTO_KE_KEYDERIVATION_HSMKEYINDEX, HsmKeyIndex, sizeof(HsmKeyIndex)))

- If both the CRYPTO_KE_KEYDERIVATION_PASSWORD and the CRYPTO_KE_KEYDERIVATION_HSMKEYINDEX have valid values set, then pbkdf2 is performed based on the HSM Key Index that is set.

	Document No.	AUTOSAR HaeModule Crypto Driver User Manual	Revision Date	'24. 04
	-		Revision No.	1.03b

6.1.15.CRYPTO_KEYSETVALID

Crypto_PrimitiveInfoType

Service	Algorithm Family	Algorithm Secondary Family	Algorithm Mode
CRYPTO_KEYSETVALID	CRYPTO_ALGOFAM_NOT_SET	CRYPTO_ALGOFAM_NOT_SET	CRYPTO_ALGOMODE_NOT_SET

Crypto_JobType

Type	Parameter	Description
uint32	cryptoKeyId	Target key

	Document No.	AUTOSAR HaeModule Crypto Driver User Manual	Revision Date	'24. 04
	-		Revision No.	1.03b

6.2. How to Add New Users to Crypto Primitive

Describes the case of adding a new Crypto Primitive to "Ecud_Crypto_76_HaeModule.arxml".

Suppose you have created a new Primitive named "HaeCryptoLib_Primitive_Sha256" in the CryptoLib Object's CryptoPrimitives as follows:

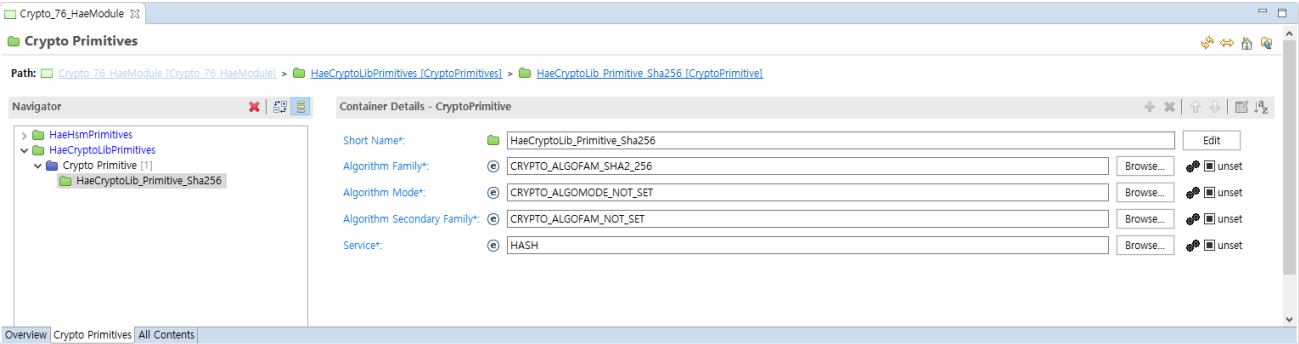


Figure 7. New CryptoPrimitive Addition

The new Primitives are added to the "Crypto_76_HaeModule_Cfg.c" file generated by the Crypto Generator tool as follows:

```
const CryptoPrimitive HaeCryptoLib_CryptoPrimitives[] =
{
    {
        CRYPTO_HASH,
        CRYPTO_ALGOFAM_SHA2_256,
        CRYPTO_ALGOFAM_NOT_SET,
        CRYPTO_ALGOMODE_NOT_SET,
        (Crypto_PrimitiveProcess) HaeCryptoLib_Primitive_Sha256
    },
};
```

When the executable function associated with the new Primitive is generated, it is created with the name of the primitive. Therefore, the user must implement a new primitive executable function of type "Crypto_PrimitiveProcess".

	Document No.	AUTOSAR HaeModule Crypto Driver User Manual	Revision Date	'24. 04
	-		Revision No.	1.03b

6.3. Setting up the Crypto MainFunction for the Asynchronous Job

The Crypto MainFunction must be configured to be executed at regular intervals in order to process Asynchronous job. However, this is not necessary if you only need to process synchronous jobs.

8.4 Scheduled functions

8.4.1.1 Crypto_MainFunction

The `Crypto_MainFunction()` is necessary for asynchronous job processing. For synchronous job processing providing the main function is optional.

[SWS_Crypto_91012] [

Service name:	Crypto_MainFunction
Syntax:	void Crypto_MainFunction(void)
Service ID[hex]:	0x0c
Description:	If asynchronous job processing is configured and there are job queues, the function is called cyclically to process queued jobs.
Available via:	SchM_Crypto.h

The Crypto MainFunction for HaeModule is defined in "Bswmd_Crypto_76_HaeModule.arxml". If you have an asynchronous job and a job queue set up, you need to configure this function to run at regular intervals.

Each Object in HaeModule has a MainFunction created to handle Asynchronous Jobs. Therefore, for normal job processing, each MainFunction should be set to perform at regular intervals.

No.	Object	MainFunction
1	HaeHsm	Crypto_HaeHsm_MainFunction
2	HaeCryptoLib	Crypto_HaeCryptoLib_MainFunction

ticket 4. Object star Crypto MainFunction

The MainFunction function is created in the "Crypto_76_HaeModule_Cfg.c" file for each Object at the time of Generate. Therefore, if an Object is deleted because it is not used, the MainFunction of that Object will not be created.

	Document No.	AUTOSAR HaeModule Crypto Driver User Manual	Revision Date	'24. 04
	-		Revision No.	1.03b

7. Key Management

7.1. Crypto Key Element Settings

The CryptoKeyElement configuration format only supports CRYPTO_KE_FORMAT_BIN_OCTET.

7.1.1. Setting up the HSM Crypto Key Element

The HSM manages the Crypto Key separately internally. Depending on the specification, the Key table provided by the HSM is different. For more information, please refer to the "Key Management" section of the user manual document that is distributed with the HSM module.

In order to associate the key inside the HSM with the key element of the HaeModule Crypto Driver, a special configuration is required. Here's how to set it up.

Type	AES Key			RSA Key			ECC p256r1 Key			ECC p521r1 Key			ECC ED448 Key [Sign/Verify]			ECC X448 Key [ECDH]		
	Key Index	영역	목적	Key Index	영역	목적	Key Index	영역	목적	Key Index	영역	목적	Key Index	영역	목적	Key Index	영역	목적
HKMC PSK Pre Shared Key	#1~#5	HKMC PSK	OTA(aSIMS)	#1	HKMC PSK	aSIMS	#1	HKMC PSK	aSIMS	미제공	미제공	미제공	미제공	미제공	미제공	미제공	미제공	미제공
	#6~#10		OTA(FST)															
	#11~#15		FoD															
	#16~#50		TBD															
Tier PSK Pre Shared Key	#101	Tier PSK	TBD	#101	Tier PSK	TBD	#101	Tier PSK	TBD	미제공	미제공	미제공	미제공	미제공	미제공	미제공	미제공	미제공
	#102			#102														
	#103			#103														
	#104			#104														
	#105			#105														
UDK (User Defined Key)	#201	UDK-AES128 (Updatable)	TBD	#201	UDK	TBD	#201	UDK	TBD	#201	UDK	TBD	#201	UDK	TBD	#201	UDK	TBD
	#202			#202			#202			#202			#202					
	#203			UDK-AES256 (Updatable)	미제공	미제공	미제공	미제공	미제공	미제공	미제공	미제공	미제공	미제공	미제공	미제공	미제공	미제공
	#204																	
	#205																	
	#206																	
	#207																	
	#208																	
HOST TEMP KEY	#301	RAM KEY AES 128	TBD	#301	RAM KEY	TBD	#301	RAM KEY	TBD	#301	RAM KEY	TBD	#301	RAM KEY	TBD	#301	RAM KEY	TBD
	#302	RAM KEY AES 256																

Table 5. HSM Key Table

Crypto Key Element and HSM Key's For the connection CryptoKeyElement with "Hsm Key Type"and Hsm Key Index” Added a setting. Hsm Key Type and Key Index are "Table 5. HSM Key Table”As shown in HSM As stated in the module's user manual, Refer to the Key Type and Index.

	Document No.	AUTOSAR HaeModule Crypto Driver User Manual	Revision Date	'24. 04
	-		Revision No.	1.03b

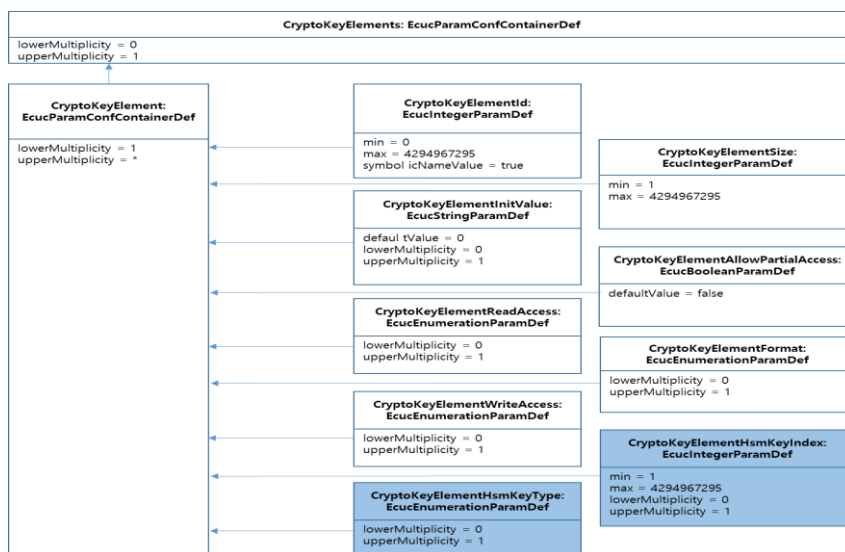


Figure 8. HaeModule CryptoKeyElement Container

Below is an example of a UDK #201과 connection of an HSM's AES Key.

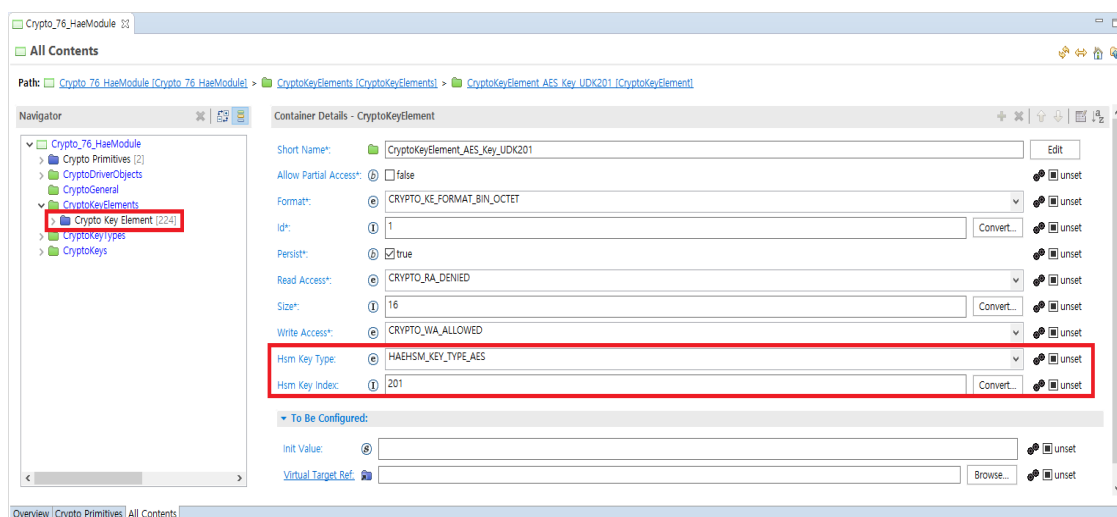


Figure 9. Setting up the HSM Crypto Key Element

If the Crypto Key Element is not related to the HSM Key, do not set the "Hsm Key Type" and "Hsm Key Index". However, if you set it up, the HaeModule Crypto Driver will try to configure the set Key Element using the HSM Key Management API, which will cause a malfunction.

	Document No.	AUTOSAR HaeModule Crypto Driver User Manual	Revision Date	'24. 04
	-		Revision No.	1.03b

7.1.2. HSM Crypto Key Element Attributes

If the Crypto Key Element is associated with the HSM Key, it is actually stored in the HSM through the HSM Driver when the "Crypto_KeySetValid" Interface is called. For example, in the case of AES Key, the Crypto Key Element is an Init with a CRYPTO_KE_CIPHER_IV(5). The Element, which is a vector, is stored in the HOST's RAM and is not stored in the HSM when the "Crypto_KeySetValid" Interface is called.

In the case of keys stored in HSMs, they can only be referenced by the Key Index and can never be read. Therefore, the CryptoKeyElement's CryptoKeyElementReadAccess must be set as a CRYPTO_RA_DENIED property. In addition, only UDK (User Defined Key) and HTK (Host Temp Key) are the permissions of CryptoKeyElementWriteAccess enabled.

In summary, the HSM-specific CryptoKeyElement has the following characteristics:

1. Keys stored and managed in HSMs can never be read by the HOST and can only be referenced by the Key Index.
2. The CryptoKeyElementReadAccess of all CryptoKeyElements must be set to the CRYPTO_RA_DENIED property.
3. The CryptoKeyElementAllowPartialAccess of all CryptoKeyElements must be set to FALSE.
4. The CryptoKeyElementWriteAccess of Tier Key, PSK must be set to the CRYPTO_WA_DENIED property.
5. The CryptoKeyElementWriteAccess of the User Defined Key (UDK) and Host Temp Key (HTK) must be set as CRYPTO_WA_ALLOWED attributes.
6. CryptoKeyElementInitValue is the AES Key's IV, UDK, and HOST TEMP KEY types can be set. At the beginning of the driver, the CryptoKeyElementInitValue value is applied only if the key value is not set before.
7. If the UDK and HTK already have a set value, the CryptoKeyElementInitValue will not be renewed. Therefore, if the value of CryptoKeyElementInitValue is changed in the middle of the year, it will not be renewed. In this case, you need to use the "Crypto_KeyElementSet" and "Crypto_KeySetValid" interfaces to configure UDK and HTK directly.

	Document No.	AUTOSAR HaeModule Crypto Driver User Manual	Revision Date	'24. 04
	-		Revision No.	1.03b

7.2. How to set CryptoKeyElementValue

7.2.1. How to enter with CryptoKeyElementInitValue

CryptoKeyElementInitValue is initialized to CryptoKeyElementValue at CryptoDriver initialization if the user has preset the correct value. In principle, the array rules of the C language standard are applied to the 1-byte Hex value. Here's an example of a valid input value:

(e.g.) 0x16, 0x15, 0x7e, 0x2b, 0xa6, 0xd2, 0xae, 0x28, 0x88, 0x15, 0xf7, 0xab

Alternatively, the following input formats are also possible, but not recommended:

(Example) 16 15 7e 2b a6 d2 ae 28 88 15 f7 ab

For Word that is more than 2 bytes, the Endian format of HOST is followed. For example, if the HOST is Little Endian, enter the following:

(Example) 210309: 0x85, 0x35, 0x03, 0x00

7.2.2. How to enter with the Crypto_KeyElementSet API


Crypto_KeyElementSet allows you to change the CryptoKeyElementValue during runtime. Crypto_KeyElementValid must be set to a valid state before the key can be utilized.

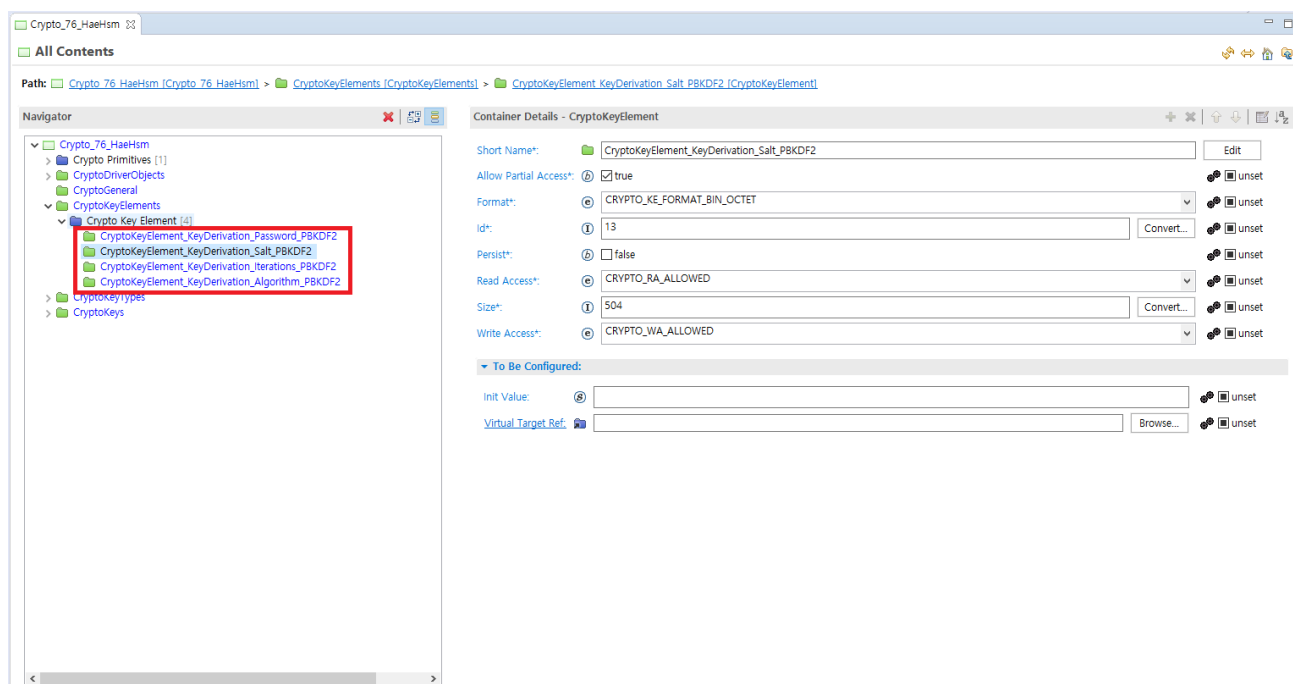
7.3. How to Add a New CryptoKey

Describes when a new CryptoKey is added to "Ecud_Crypto_76_HaeModule.arxml". Key derivation using the PBKDF2 algorithm is explained with an example. If you look at the 'SWS_Csm_01022' of the AUTOSAR_SWS_CryptoServiceManager manual, you will need four key elements for key derivation as follows.

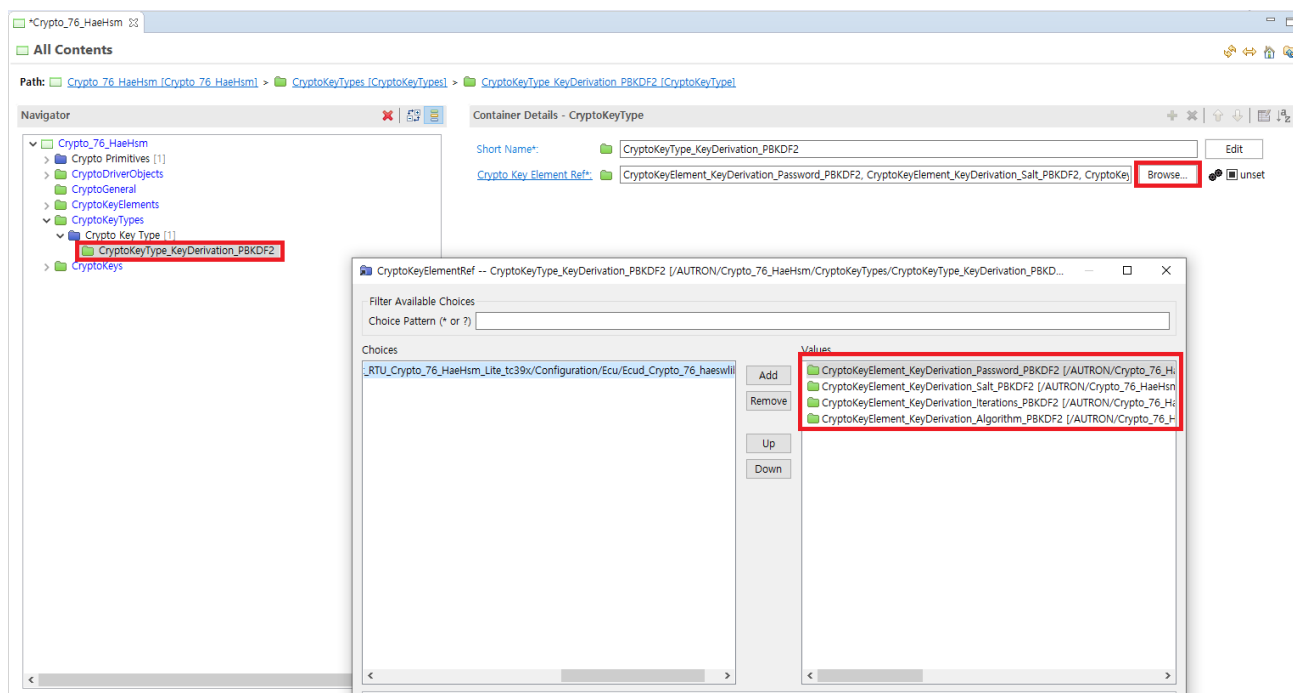
Key Derivation	Password	CRYPTO_KE_KEYDERIVATION_PASSWORD	1	x
	Salt	CRYPTO_KE_KEYDERIVATION_SALT	13	
	Iterations	CRYPTO_KE_KEYDERIVATION_ITERATIONS	14	
	Algorithm	CRYPTO_KE_KEYDERIVATION_ALGORITHM	15	

Create 4 Key Elements in CryptoKeyElements as follows:

	Document No.	AUTOSAR HaeModule Crypto Driver User Manual	Revision Date	'24. 04
	-		Revision No.	1.03b

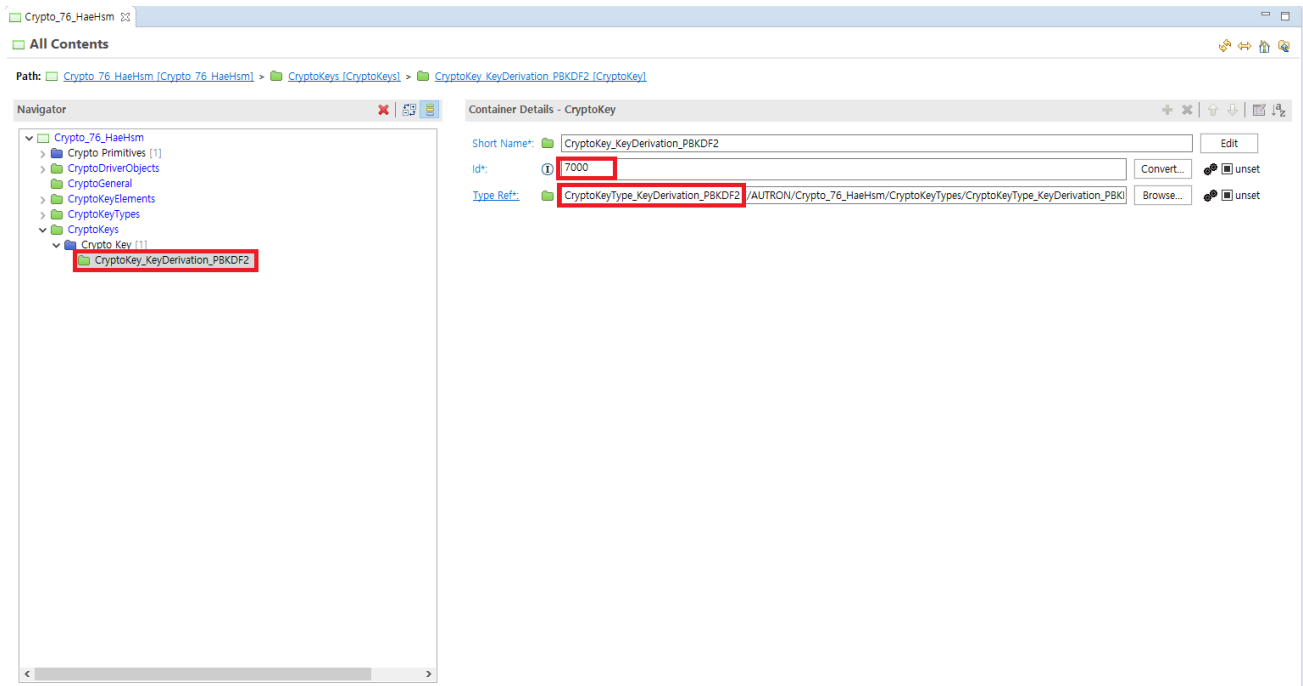


Create a 'CryptoKeyType_KeyDerivation_PBKDF2' for CryptoKeyTypes. Then, add the 4 Key Elements created above to the 'Crypto Key Element Ref'.



Create a 'CryptoKey_KeyDerivation_PBKDF2' in the CryptoKeys. Then, add the 'CryptoKeyType_KeyDerivation_PBKDF2' created above to the 'Type Ref'. Here, the CryptoKeyId is set to 7000.

	Document No.	AUTOSAR HaeModule Crypto Driver User Manual	Revision Date	'24. 04
	-		Revision No.	1.03b



The user can manipulate or reference the Key Element of the Key Derivation added here with a CryptoKeyId value of 7000.

<div> <div>HYUNDAI</div> <div>MOTOR GROUP</div> </div>	Document No.	AUTOSAR HaeModule Crypto Driver User Manual	Revision Date	'24. 04
	-		Revision No.	1.03b

8. Crypto Configuration

Describes the containers and parameters of the Crypto module. For detailed parameter meanings, please refer to the "Specification of Crypto Driver" document of AUTOSAR , in particular the "Configuration specification".

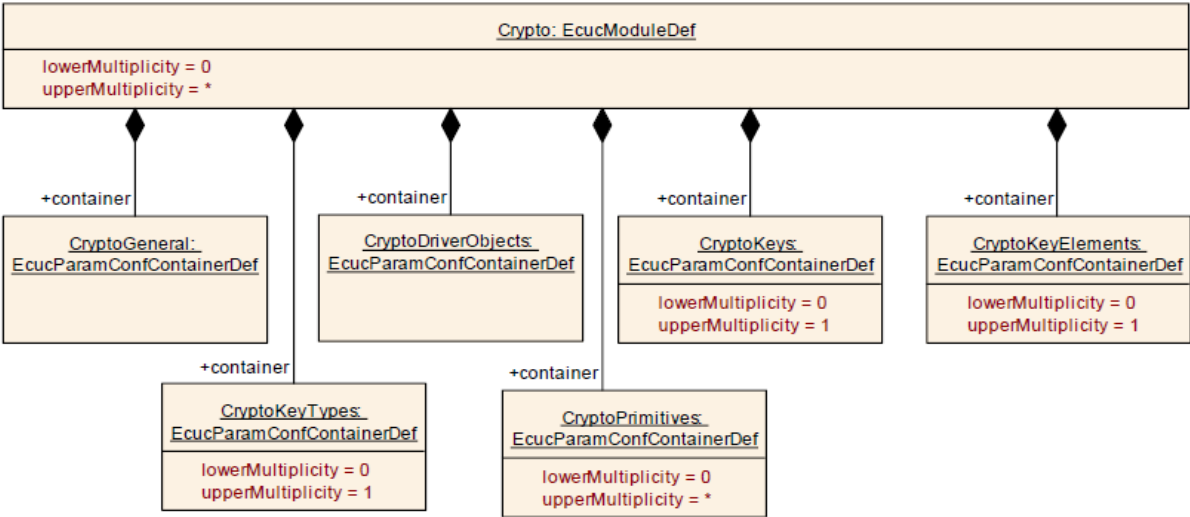



Figure 10. AUTOSAR Crypto Container Structure

The ECU description of the provided HaeModule Crypto Driver follows the AUTOSAR Crypto Container structure.

In addition, the 'Ecud_Crypto_76_HaeModule.arxml' file contains the default values required to use the HSM security module.

	Document No.	AUTOSAR HaeModule Crypto Driver User Manual	Revision Date	'24. 04
	-		Revision No.	1.03b

8.1. CryptoGeneral

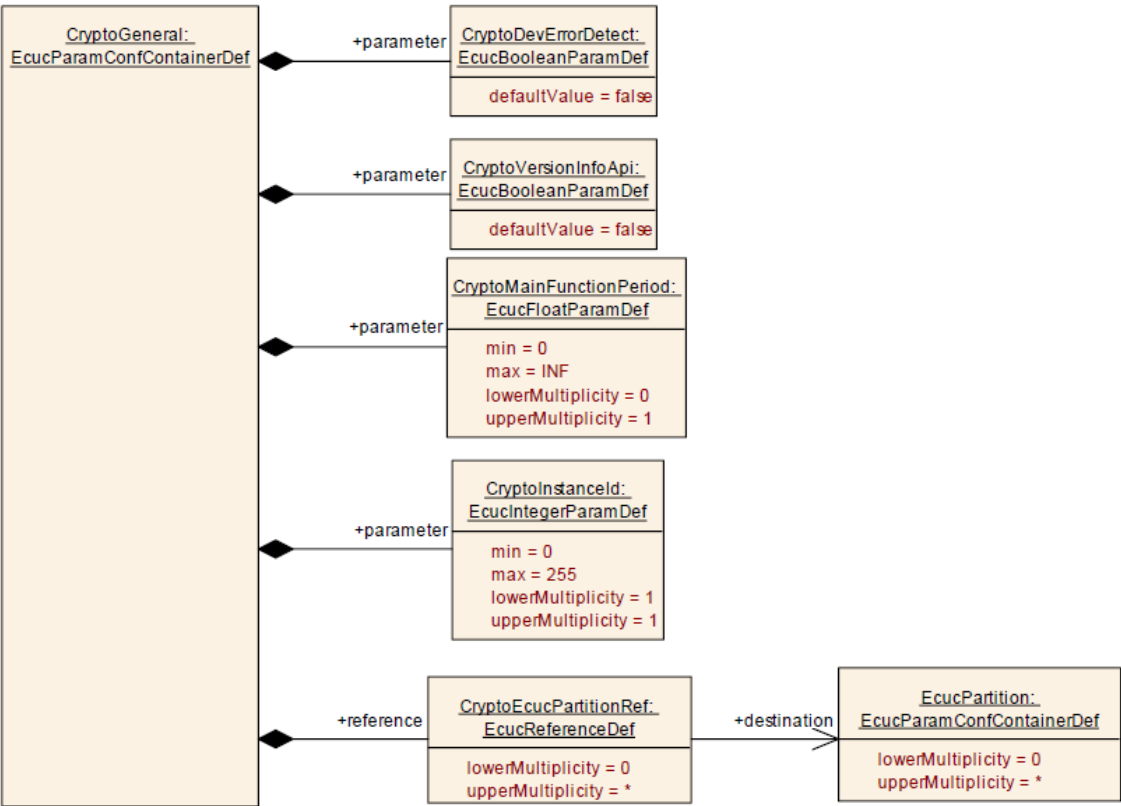


Figure 11. AUTOSAR CryptoGeneral Container

The default settings of the CryptoGeneral Container of the HaeModule Crypto Driver are shown in the table below. All settings can be changed to suit the user's environment.

Name	CryptoDevErrorDetect	CryptoVersionInfoApi	CryptoMainFunctionPeriod	CryptoInstancelId	CryptoEcucPartitionRef
CryptoGeneral	TRUE	TRUE	0.1	1	NULL

Table 6. HaeModule CryptoGeneral Container Contents

8.2. CryptoDriverObjects

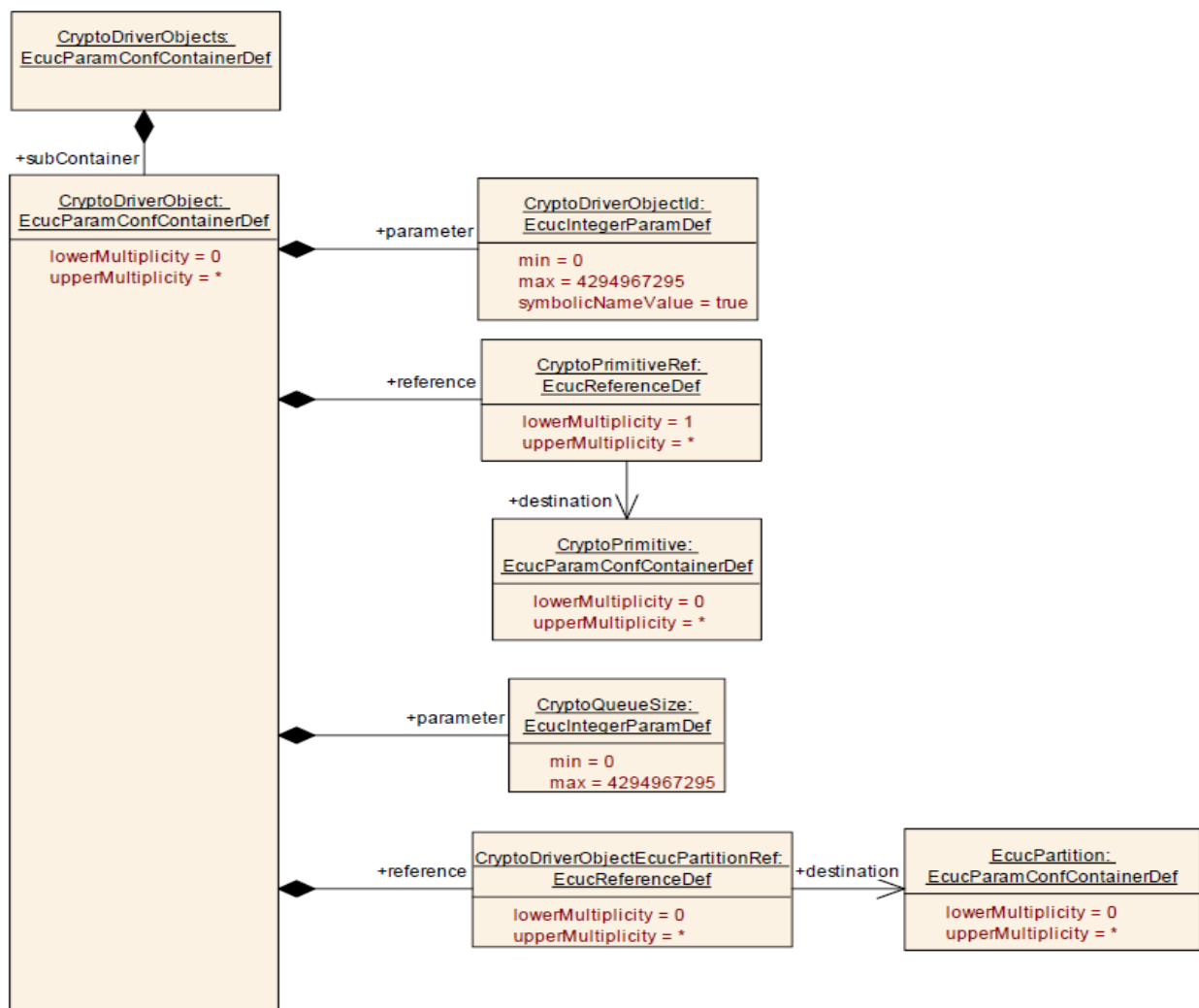


Figure 12. AUTOSAR CryptoDriverObject Container

The default settings for the CryptoDriverObject Container of the HaeModule Crypto Driver are shown in the table below. The user can change the following items according to the usage environment.

- CryptoQueueSize
- CryptoDriverObjectEcucPartitionRef

Name	CryptoDriverObjectId	CryptoQueueSize	CryptoPrimitiveRef	CryptoDriverObjectEcucPartitionRef
HaeHsm	0	10	HaeHsm_CryptoPrimitives	NULL
HaeCryptoLib	1	10	HaeCryptoLib_CryptoPrimitives	NULL

Table 7. HaeModule CryptoDriverObject Container Contents

	Document No.	AUTOSAR HaeModule Crypto Driver User Manual	Revision Date	'24. 04
	-		Revision No.	1.03b

8.3. CryptoPrimitives

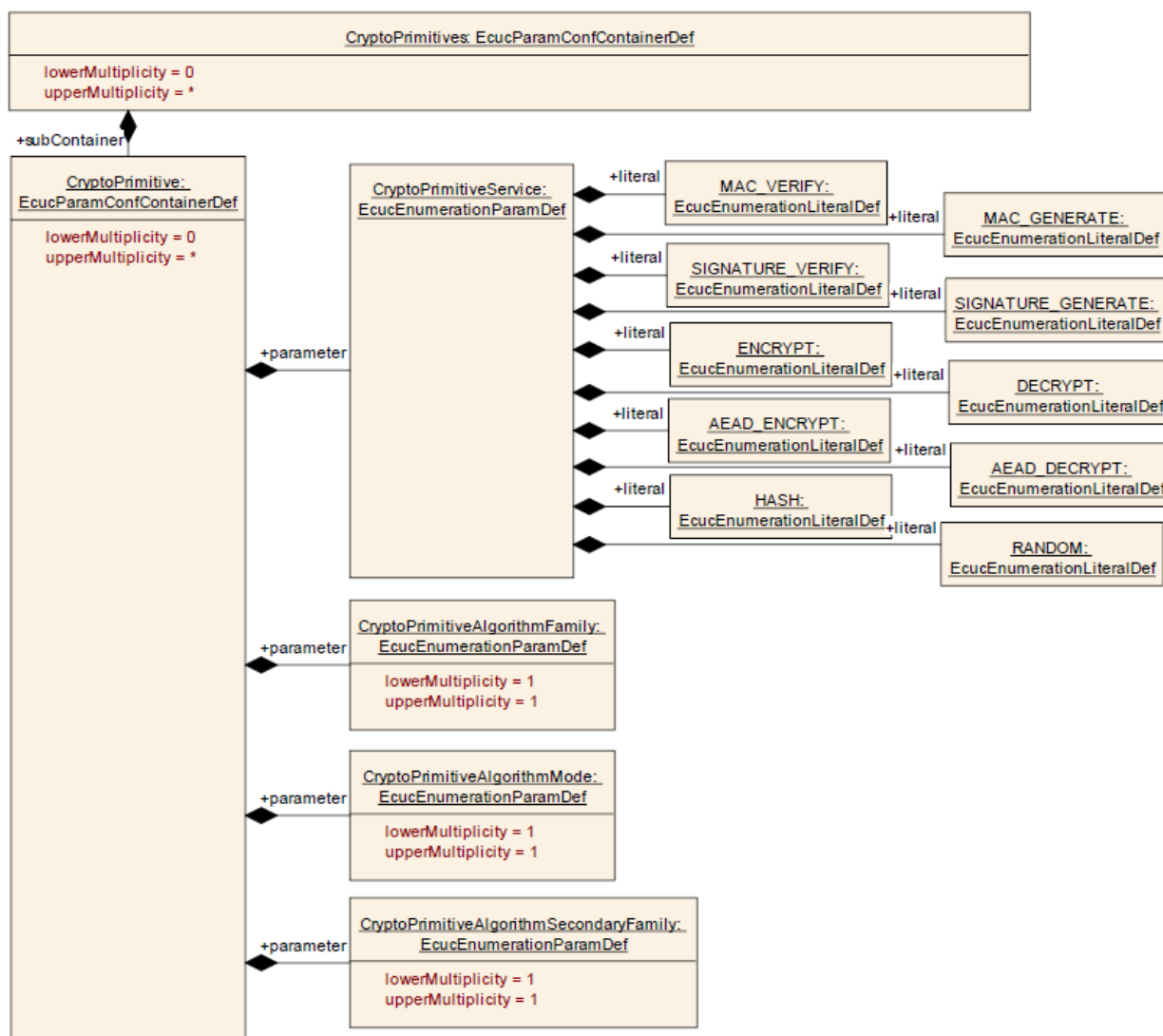


Figure 13. AUTOSAR CryptoPrimitive Container

HaeModule Crypto Driver's CryptoPrimitive Container is a combination of an HSM module and a CryptoLib. It services the Crypto algorithm provided by the module. For detailed service-specific parameter settings, please refer to this manual. "6.1. Job Primitive Settings by Crypto Algorithm" Please refer to the contents. And if the user adds the service as needed, the "6.2. How to Add New Users to Crypto Primitive" It is possible to refer to the content and add it.

<div> <div>HYUNDAI</div> <div>MOTOR GROUP</div> </div>	Document No.	AUTOSAR HaeModule Crypto Driver User Manual	Revision Date	'24. 04
	-		Revision No.	1.03b

8.4. CryptoKeys

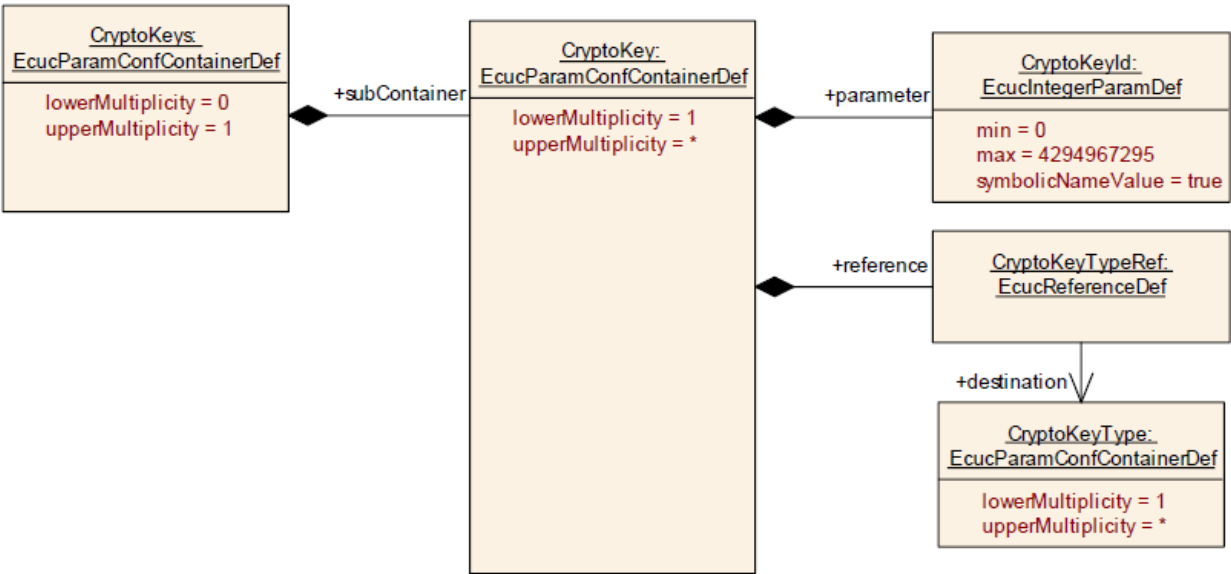


Figure 14. AUTOSAR CryptoKey Container

The CryptoKey Container of the HaeModule Crypto Driver can be associated with the keys managed within the HSM. **At this time, KeyId is duplicated If you set it, it will malfunction.** HSM In addition to the key If you would like to add a user key, please use the “7.3. How to Add a New CryptoKey ” Please refer to the contents.

	Document No.	AUTOSAR HaeModule Crypto Driver User Manual	Revision Date	'24. 04
	-		Revision No.	1.03b

8.5. CryptoKeyTypes

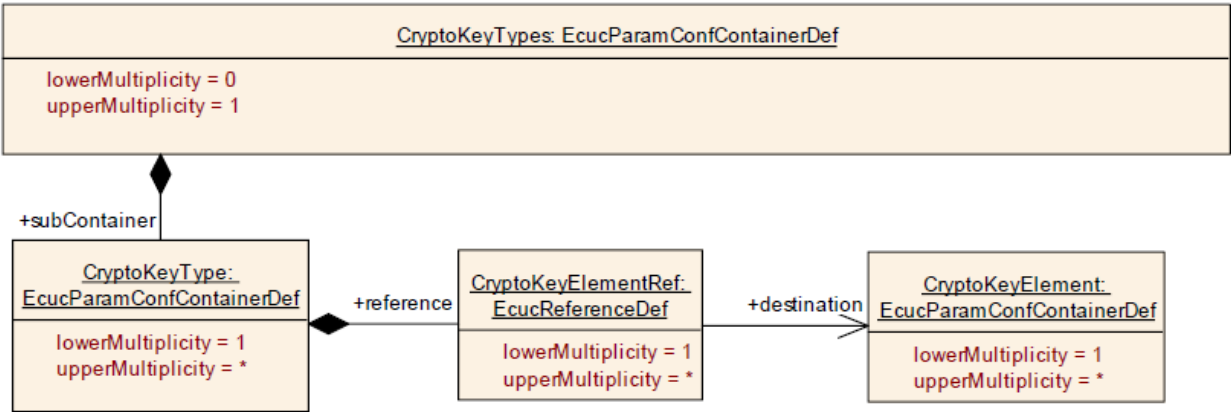
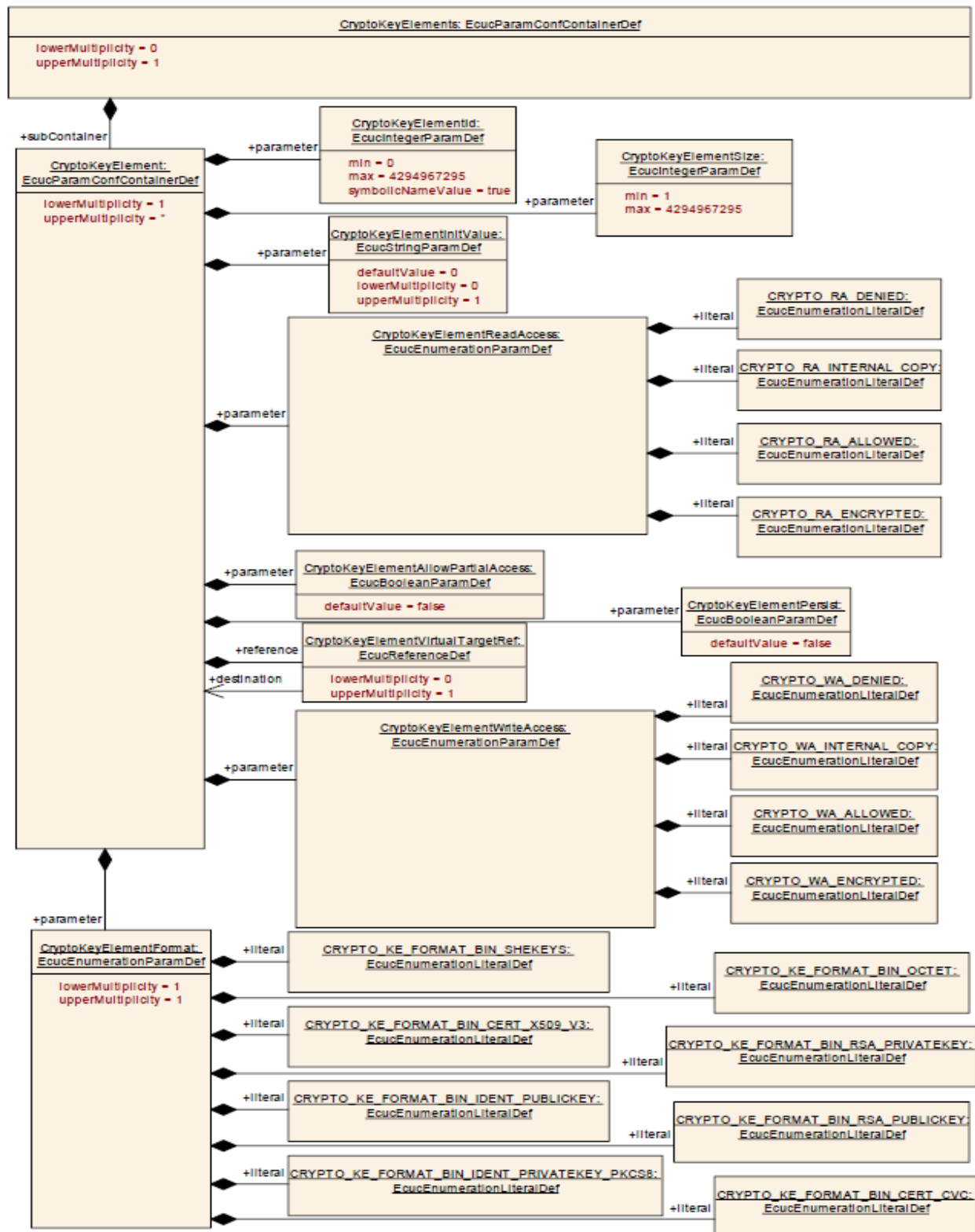


Figure 15. AUTOSAR CryptoKeyType Container

The CryptoKeyType Container of the HaeModule Crypto Driver has the necessary values set by default. The AES key requires an Initialization Vector (IV). This IV Element is created and managed on the HOST's RAM. For example, CryptoKeyType_AES_PSK1 has two Elements as references. One has a CryptoKeyElement_AES_Key_PSK1, which is an element stored in an HSM, and a CryptoKeyElement_AES_IV_PSK1, which is an element that is managed on HOST RAM.

	Document No.	AUTOSAR HaeModule Crypto Driver User Manual	Revision Date	'24. 04
	-		Revision No.	1.03b

8.6. CryptoKeyElements



	Document No.	AUTOSAR HaeModule Crypto Driver User Manual	Revision Date	'24. 04
	-		Revision No.	1.03b

Figure 16. AUTOSAR CryptoKeyElement Container

HaeModule Crypto Driver's CryptoKeyElement Container is a The values required for use are set by default. For more information on CryptoKeyElement in HSMs, please refer to this manual “7.1.2. HSM Crypto Key Element Attributes” Please refer to the

	Document No.	AUTOSAR HaeModule Crypto Driver User Manual	Revision Date	'24. 04
	-		Revision No.	1.03b

9. Crypto Driver Porting Guide

9.1. mobilgene Classic Platform

For a porting guide in the Mobile Gene Classic environment, please refer to the following site. The latest guide will be updated, so be sure to check back often.

- Platform version: R4.x

https://swpfaq.hyundai-autoever.com/display/MCF/%5BHAE_HSM%5D%5BR4X%5D%5BIM%5D+IM+of+Crypto_76_HaeHsm

- Platform version: R4.4

https://swpfaq.hyundai-autoever.com/display/MCF/%5BHAE_HSM%5D%5BR44%5D%5BIM%5D+IM+of+Crypto_76_HaeHsm

9.2. AUTOSAR Platform

Refer to your platform's manual and AUTOSAR Crypto's manual before installing.

	Document No.	AUTOSAR HaeModule Crypto Driver User Manual	Revision Date	'24. 04
	-		Revision No.	1.03b

9.3. Memory Section

9.3.1. Link Script Section

The HaeModule Crypto Driver has the following memory sections: Place the section in the Link Script .

Sections are defined in the following files:

Crypto_76_HaeModule_R44/delivery/inc/ **Crypto_76_HaeModule_MemMap.h**

category	Section Name	memory	explanation
rodata	CRYPTO_76_HAEMODULE_ROM_CONST_UNSPECIFIED	Flash	Read Only Data
bss	CRYPTO_76_HAEMODULE_RAM_VAR_CLEARED_UNSPECIFIED	RAM	Variable data with no initial value. Set it to the 0x00 value at boot time.
data	CRYPTO_76_HAEMODULE_RAM_VAR_INIT_UNSPECIFIED	RAM	Variable data with initial values. The default value is stored in Flash and copied to the address of the RAM section at boot.
text	CRYPTO_76_HAEMODULE_SEC_CODE	Flash	Program Code

The following is an example of the Tasking compiler. Each compiler has a different link script, and the section names may be slightly different. Refer to the compiler manual for the exact details. However, the general content is similar, and it is thought that it can be set in other compilation environments based on the explanation below.

rodata

If you want it to be placed in the same place as the existing rodata section, check the Link Script to see if one of the following section names exists:

- select ".rodata.*_ROM_CONST_*";
- select ".rodata*";

If you want to place it in an existing or new section, specify the desired section location as follows:

- select ".rodata. CRYPTO_76_HAEMODULE_ROM_CONST_UNSPECIFIED";

bss

If you want it to be placed in the same place as an existing bss section, check the Link Script to see if

	Document No.	AUTOSAR HaeModule Crypto Driver User Manual	Revision Date	'24. 04
	-		Revision No.	1.03b

one of the following section names exists:

- `select ".bss.*_RAM_VAR_CLEARED_*";`
- `select ".bss*";`

If you want to place it in an existing or new section, specify the desired section location as follows:

- `select ".bss.CRYPTO_76_HAEMODULE_RAM_VAR_CLEARED_UNSPECIFIED";`

data

If you want it to be placed in the same place as the existing data section, check the Link Script to see if one of the following section names exists:

- `select ".data.*_RAM_VAR_INIT_*";`
- `select ".data*";`

If you want to place it in an existing or new section, specify the desired section location as follows:

- `select ".data.CRYPTO_76_HAEMODULE_RAM_VAR_INIT_UNSPECIFIED";`

text

If you want it to be placed in the same place as an existing text section, check the Link Script to see if one of the following section names exists:

- `select ".text.*_CODE";`
- `select ".text*";`

If you want to place it in an existing or new section, specify the desired section location as follows:

- `select ".text.CRYPTO_76_HAEMODULE_SEC_CODE";`

9.3.2. Platform memory map file

In the case of the Mobilzine platform, there is a 'MemMap.h' file, and there may be a separate 'User_MemMap.h' file depending on the version.

	Document No.	AUTOSAR HaeModule Crypto Driver User Manual	Revision Date	'24. 04
	-		Revision No.	1.03b

If a User_MemMap.h file exists (R4.4 version)

Add the following to the file:

```
#if defined (SECTION_NOT_FOUND)
    #undef SECTION_NOT_FOUND
    #include "Crypto_76_HaeModule_MemMap.h"
#endif
```

If only the MemMap.h file exists

Add the following to the end of the file:

```
#if defined (SECTION_NOT_FOUND)
    #undef SECTION_NOT_FOUND
    #include "Crypto_76_HaeModule_MemMap.h"
#endif
```

	Document No.	AUTOSAR HaeModule Crypto Driver User Manual	Revision Date	'24. 04
	-		Revision No.	1.03b

Appendix. Technical Support Guide

Technical support is available through Redmine.

For detailed procedures, please refer to the user manual that comes with the HSM security module.