Document Title	Specification of Cryptography
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	883

Document Status	published
Part of AUTOSAR Standard	Adaptive Platform
Part of Standard Release	R25-11

Document Change History

Date	Release	Changed by	Description
2025-11-27	R25-11	AUTOSAR Release Management	• Reworked the entire Crypto, Keys, and Certificates handling APIs
2024-11-27	R24-11	AUTOSAR Release Management	• Improve Crypto APIs by removing default values and using out parameter instead of returning ara::core::vector • Improve Keyslot and Certificate access control • Improve UpdatesObserver functional specification • Improve Signature handling • Add Log and Trace functionality to Crypto API
2023-11-23	R23-11	AUTOSAR Release Management	• Improved Crypto API Functional specification in Chapter 7 • Improved Chapter 8. API specification • Added initial version from Chapter 5. Dependencies to other functional clusters

△

2022-11-24	R22-11	AUTOSAR Release Management	• Added: [SWS_CRYPT_40984] [SWS_CRYPT_40985] • Updated: [SWS_CRYPT_00502] [SWS_CRYPT_00503] [SWS_CRYPT_40983] [SWS_CRYPT_01820] [SWS_CRYPT_01821] [SWS_CRYPT_00919] [SWS_CRYPT_03303] [SWS_CRYPT_03305] [SWS_CRYPT_04204] [SWS_CRYPT_01821] [SWS_CRYPT_00919] • Updated API: [SWS_CRYPT_40981] [SWS_CRYPT_20746] [SWS_CRYPT_21115] [SWS_CRYPT_21312] [SWS_CRYPT_22711] [SWS_CRYPT_23012] [SWS_CRYPT_23013] [SWS_CRYPT_23014] • Editorial changes
2021-11-25	R21-11	AUTOSAR Release Management	• Reworked SWS items in chapter 7 to improve testability • Improved traceability between chapters 7 and 8
2020-11-30	R20-11	AUTOSAR Release Management	• Rewrote the document to align with AUTOSAR standard • Update of Crypto API according to WG-SEC feedback
2019-03-29	R19-03	AUTOSAR Release Management	• "Direct" prefix of Crypto API is removed, because now it is single • All bugs found after R18-03 are fixed • Crypto API is converted for usage of basic `ara::core` types • Crypto API is converted for support of the "Exception-less" approach • Detalization of Crypto API specification is extended
2018-08-20	R18-10	AUTOSAR Release Management	• Removed crypto API introduced in release 17-10

▽

△

2018-03-29	R18-03	AUTOSAR Release Management	• Crypto API introduced at previous release is renamed to Modeled API, chapter 7 is updated • Added specification of additional Direct Crypto API (chapter 9)
2017-10-27	R17-10	AUTOSAR Release Management	• Initial release

Disclaimer

This work (specification and/or software implementation) and the material contained in it, as released by AUTOSAR, is for the purpose of information only. AUTOSAR and the companies that have contributed to it shall not be liable for any use of the work.

The material contained in this work is protected by copyright and other types of intellectual property rights. The commercial exploitation of the material contained in this work requires a license to such intellectual property rights.

This work may be utilized or reproduced without any modification, in any form or by any means, for informational purposes only. For any other purpose, no part of the work may be utilized or reproduced, in any form or by any means, without permission in writing from the publisher.

The work has been developed for automotive applications only. It has neither been developed, nor tested for non-automotive applications.

The word AUTOSAR and the AUTOSAR logo are registered trademarks.

Table of Contents

1 Introduction and functional overview 22

2 Acronyms and Abbreviations 23

3 Related documentation 27
3.1 Input documents & related standards and norms . . . 27
3.2 Further applicable specification . . . 28

4 Constraints and assumptions 29
4.1 Assumptions on the Implementation of Crypto . . . 29
4.1.1 Process Isolation . . . 29
4.1.2 Usage of Cryptographic Modules . . . 29
4.2 Known limitations . . . 29

5 Dependencies to other functional clusters 30
5.1 Provided Interfaces . . . 30
5.2 Required Interfaces . . . 31

6 Requirements Tracing 32

7 Functional specification 38
7.1 Architecture & Concepts . . . 38
7.1.1 Architecture . . . 38
7.1.2 Usage of Crypto . . . 40
7.1.2.1 Usage of Crypto by Applications . . . 40
7.1.2.2 Usage of Crypto by other Functional Clusters . . . 40
7.1.3 Access Control . . . 41
7.1.4 Usage Control . . . 43
7.1.5 Identification of Cryptographic Transformations and Key Types . . . 44
7.1.6 Concurrent Access . . . 46
7.2 Functional Cluster Lifecycle . . . 47
7.2.1 Startup . . . 47
7.2.2 Shutdown . . . 47
7.3 Crypto Provider . . . 48
7.3.1 KeySlot Access . . . 49
7.3.2 CryptoProvider Instantiation . . . 50
7.3.3 Runtime Key Instantiation and Usage . . . 51
7.3.3.1 Utility functions . . . 51
7.3.4 Key Generation . . . 51
7.3.5 Plaintext Key Management . . . 52
7.3.5.1 Plaintext Key Import . . . 52
7.3.5.2 Plaintext Key Export . . . 53
7.3.6 Protected Key Management . . . 53
7.3.6.1 Protected Key Import . . . 53

7.3.6.2 Protected Key Export . . . 54
7.3.7 Import and Export Formats . . . 54
7.3.8 Context Creation . . . 56
7.3.9 CryptoProvider Identification . . . 56
7.4 Crypto Contexts . . . 57
7.4.1 Overview . . . 57
7.4.1.1 Set Key . . . 57
7.4.1.2 Single-call API . . . 57
7.4.1.3 Streaming API . . . 57
7.4.1.4 Informational API . . . 58
7.4.1.5 Output Buffers . . . 58
7.4.1.6 Initialization Vectors . . . 58
7.4.2 Random Number Generator (RNG) . . . 58
7.4.3 Hashing . . . 60
7.4.3.1 Streaming API . . . 60
7.4.3.2 Single-call API . . . 60
7.4.3.3 Suspend and Resume Hash Calculations . . . 61
7.4.4 Message Authentication Code (MAC) . . . 61
7.4.4.1 Streaming API . . . 61
7.4.4.2 Single-call API . . . 62
7.4.5 Digital signatures . . . 62
7.4.5.1 Streaming API . . . 63
7.4.5.2 Single-call API . . . 63
7.4.6 Symmetric Encryption & Decryption . . . 63
7.4.6.1 Streaming API . . . 64
7.4.6.2 Single-call API . . . 64
7.4.7 Asymmetric Encryption & Decryption . . . 64
7.4.8 Authenticated Encryption with Associated Data . . . 65
7.4.9 Key Derivation Function (KDF) . . . 65
7.4.10 Key Wrapping . . . 66
7.4.11 Key Encapsulation Mechanism (KEM) . . . 67
7.4.12 Key Agreement . . . 67
7.5 Certificate handling (X.509 Provider) . . . 68
7.5.1 X.509 Certificate Parsing . . . 68
7.5.2 X.509 Certificate Validation . . . 70
7.5.3 X.509 Certificate Storage . . . 71
7.5.4 Certificate Signing Requests (CSR) . . . 73
7.5.5 Certificate Revocation Lists (CRL) . . . 73
7.5.6 Online Certificate Status Protocol (OCSP) . . . 73
7.6 Reporting . . . 74
7.6.1 Security Events . . . 74
7.6.2 Log Messages . . . 78
7.6.3 Violation Messages . . . 80

7.6.4 Production Errors . . . 81
8 API specification 82
8.1 PortInterface to API class binding . . . 82
8.2 Header: ara/crypto/common/base_id_types.h . . . 83
8.2.1 Non-Member Types . . . 83
8.2.1.1 Type Alias: AlgId . . . 83
8.2.1.2 Type Alias: AlgIdString . . . 84
8.2.1.3 Type Alias: AllowedUsageFlags . . . 84
8.2.1.4 Enumeration: CryptoObjectType . . . 84
8.2.1.5 Enumeration: CryptoTransform . . . 85
8.2.1.6 Type Alias: Timestamp . . . 85
8.2.2 Global Variables . . . 86
8.2.2.1 kAllowDataDecryption . . . 86
8.2.2.2 kAllowDataEncryption . . . 86
8.2.2.3 kAllowDerivedPlaintextExport . . . 87
8.2.2.4 kAllowHashCalculation . . . 87
8.2.2.5 kAllowKeyAgreement . . . 88
8.2.2.6 kAllowKeyDerivation . . . 88
8.2.2.7 kAllowKeyUnwrapping . . . 88
8.2.2.8 kAllowKeyWrapping . . . 89
8.2.2.9 kAllowMac . . . 89
8.2.2.10 kAllowPersist . . . 90
8.2.2.11 kAllowPlaintextExport . . . 90
8.2.2.12 kAllowRngInit . . . 90
8.2.2.13 kAllowSecuredExport . . . 91
8.2.2.14 kAllowSigGeneration . . . 91
8.2.2.15 kAllowSigVerification . . . 92
8.2.3 Struct: SecureCounter . . . 92
8.2.3.1 Public Member Variables . . . 93
8.2.3.1.1 mLSQW . . . 93
8.2.3.1.2 mMSQW . . . 93
8.3 Header: ara/crypto/common/crypto_error_domain.h . . . 94
8.3.1 Non-Member Types . . . 94
8.3.1.1 Enumeration: CryptoErrc . . . 94
8.3.2 Non-Member Functions . . . 95
8.3.2.1 Other . . . 95
8.3.2.1.1 GetCryptoErrorDomain . . . 95
8.3.2.1.2 MakeErrorCode . . . 96
8.3.3 Class: CryptoErrorDomain . . . 96
8.3.3.1 Public Member Types . . . 97
8.3.3.1.1 Type Alias: Errc . . . 97
8.3.3.1.2 Type Alias: Exception . . . 97
8.3.3.2 Public Member Functions . . . 98

8.3.3.2.1 Special Member Functions . . . 98
8.3.3.2.1.1 Default Constructor . . . 98
8.3.3.2.2 Member Functions . . . 98
8.3.3.2.2.1 Message . . . 98
8.3.3.2.2.2 Name . . . 99
8.3.3.2.2.3 ThrowAsException . . . 99
8.3.4 Class: CryptoException . . . 100
8.3.4.1 Public Member Functions . . . 100
8.3.4.1.1 Constructors . . . 100
8.3.4.1.1.1 CryptoException . . . 100
8.4 Header: ara/crypto/common/entry_point.h . . . 101
8.4.1 Non-Member Functions . . . 101
8.4.1.1 Other . . . 101
8.4.1.1.1 LoadCryptoProvider . . . 101
8.4.1.1.2 LoadKeySlot . . . 101
8.4.1.1.3 LoadWriteableKeySlot . . . 102
8.4.1.1.4 LoadX509Provider . . . 103
8.5 Header: ara/crypto/common/formatter.h . . . 103
8.5.1 Non-Member Functions . . . 103
8.5.1.1 Other . . . 103
8.5.1.1.1 LoadFormatter . . . 103
8.5.2 Class: Formatter . . . 104
8.5.2.1 Public Member Types . . . 104
8.5.2.1.1 Type Alias: Sptr . . . 104
8.5.2.2 Public Member Functions . . . 105
8.5.2.2.1 Special Member Functions . . . 105
8.5.2.2.1.1 Copy Constructor . . . 105
8.5.2.2.1.2 Move Constructor . . . 105
8.5.2.2.1.3 Copy Assignment Operator . . . 106
8.5.2.2.1.4 Move Assignment Operator . . . 106
8.5.2.2.1.5 Destructor . . . 106
8.5.2.2.2 Member Functions . . . 107
8.5.2.2.2.1 Convert . . . 107
8.5.2.2.2.2 GetExportSize . . . 107
8.5.2.2.2.3 SetFormatConfiguration . . . 108
8.6 Header: ara/crypto/common/mem_region.h . . . 108
8.6.1 Non-Member Types . . . 108
8.6.1.1 Type Alias: ReadOnlyMemRegion . . . 108
8.6.1.2 Type Alias: ReadWriteMemRegion . . . 109
8.7 Header: ara/crypto/common/serializable.h . . . 109
8.7.1 Non-Member Types . . . 109
8.7.1.1 Type Alias: FormatId . . . 109
8.7.2 Global Variables . . . 110

8.7.2.1 kFormatDerEncoded . . . 110
8.7.2.2 kFormatPemEncoded . . . 110
8.7.2.3 kFormatRawValueOnly . . . 111
8.7.2.4 kProprietaryFormat . . . 111
8.7.3 Class: Serializable . . . 111
8.7.3.1 Public Member Functions . . . 112
8.7.3.1.1 Special Member Functions . . . 112
8.7.3.1.1.1 Move Constructor . . . 112
8.7.3.1.1.2 Copy Constructor . . . 112
8.7.3.1.1.3 Copy Assignment Operator . . . 113
8.7.3.1.1.4 Move Assignment Operator . . . 113
8.7.3.1.1.5 Destructor . . . 113
8.7.3.1.2 Member Functions . . . 114
8.7.3.1.2.1 ExportPublicly . . . 114
8.7.3.1.2.2 GetExportSize . . . 114
8.8 Header: ara/crypto/cryp/auth_cipher_ctx.h . . . 115
8.8.1 Class: AuthCipherCtx . . . 115
8.8.1.1 Public Member Types . . . 115
8.8.1.1.1 Type Alias: Sptr . . . 115
8.8.1.2 Public Member Functions . . . 116
8.8.1.2.1 Member Functions . . . 116
8.8.1.2.1.1 Cancel . . . 116
8.8.1.2.1.2 EncryptSingleCallWithOutputIv . . . 116
8.8.1.2.1.3 Finish . . . 118
8.8.1.2.1.4 FinishEncrypt . . . 118
8.8.1.2.1.5 GetMaxAssociatedDataSize . . . 119
8.8.1.2.1.6 GetMaximumChunkSize . . . 120
8.8.1.2.1.7 GetMaximumInputIvSize . . . 120
8.8.1.2.1.8 GetMinimumChunkSize . . . 121
8.8.1.2.1.9 GetMinimumInputIvSize . . . 121
8.8.1.2.1.10 GetOutputIvSize . . . 122
8.8.1.2.1.11 GetRequiredOutputBufferSize . . . 123
8.8.1.2.1.12 GetTagSize . . . 123
8.8.1.2.1.13 ProcessSingleCall . . . 124
8.8.1.2.1.14 ProcessSingleCallWithInputIv . . . 125
8.8.1.2.1.15 SetKey . . . 126
8.8.1.2.1.16 SetKey . . . 127
8.8.1.2.1.17 Start . . . 128
8.8.1.2.1.18 Start . . . 129
8.8.1.2.1.19 UpdateAuthenticatedData . . . 130
8.8.1.2.1.20 UpdateConfidentialData . . . 130
8.9 Header: ara/crypto/cryp/crypto_context.h . . . 131
8.9.1 Class: CryptoContext . . . 131

8.9.1.1 Public Member Functions . . . 132
8.9.1.1.1 Special Member Functions . . . 132
8.9.1.1.1.1 Copy Constructor . . . 132
8.9.1.1.1.2 Move Constructor . . . 132
8.9.1.1.1.3 Copy Assignment Operator . . . 133
8.9.1.1.1.4 Move Assignment Operator . . . 133
8.9.1.1.1.5 Destructor . . . 133
8.9.1.1.2 Member Functions . . . 134
8.9.1.1.2.1 GetAlgId . . . 134
8.10 Header: ara/crypto/cryp/crypto_provider.h . . . 134
8.10.1 Class: CryptoProvider . . . 134
8.10.1.1 Public Member Types . . . 135
8.10.1.1.1 Type Alias: Sptr . . . 135
8.10.1.2 Public Member Functions . . . 135
8.10.1.2.1 Special Member Functions . . . 135
8.10.1.2.1.1 Move Constructor . . . 135
8.10.1.2.1.2 Copy Constructor . . . 136
8.10.1.2.1.3 Move Assignment Operator . . . 136
8.10.1.2.1.4 Copy Assignment Operator . . . 136
8.10.1.2.1.5 Destructor . . . 137
8.10.1.2.2 Member Functions . . . 137
8.10.1.2.2.1 CreateAuthCipherCtx . . . 137
8.10.1.2.2.2 CreateDecryptorPrivateCtx . . . 138
8.10.1.2.2.3 CreateEncryptorPublicCtx . . . 138
8.10.1.2.2.4 CreateHashFunctionCtx . . . 139
8.10.1.2.2.5 CreateKeyAgreementPrivateCtx . . . 139
8.10.1.2.2.6 CreateKeyDecapsulatorPrivateCtx . . . 140
8.10.1.2.2.7 CreateKeyDerivationFunctionCtx . . . 141
8.10.1.2.2.8 CreateKeyEncapsulatorPublicCtx . . . 141
8.10.1.2.2.9 CreateMessageAuthnCodeCtx . . . 142
8.10.1.2.2.10 CreateProtectedKeyImportCtx . . . 143
8.10.1.2.2.11 CreateRandomGeneratorCtx . . . 143
8.10.1.2.2.12 CreateSignerPrivateCtx . . . 144
8.10.1.2.2.13 CreateSymmetricCipherCtx . . . 144
8.10.1.2.2.14 CreateSymmetricKeyWrapperCtx . . . 145
8.10.1.2.2.15 CreateVerifierPublicCtx . . . 146
8.10.1.2.2.16 ExportSecuredKey . . . 146
8.10.1.2.2.17 GeneratePrivateKey . . . 147
8.10.1.2.2.18 GenerateRandomData . . . 148
8.10.1.2.2.19 GenerateSymmetricKey . . . 149
8.10.1.2.2.20 GetDefaultAllowedUsage . . . 149
8.10.1.2.2.21 GetProviderID . . . 150
8.10.1.2.2.22 GetProviderIdSize . . . 151

8.10.1.2.2.23 GetSecureCounter . . . 151
8.10.1.2.2.24 ImportPrivateKey . . . 152
8.10.1.2.2.25 ImportPublicKey . . . 152
8.10.1.2.2.26 ImportSymmetricKey . . . 153
8.10.1.2.2.27 LoadPrivateKey . . . 154
8.10.1.2.2.28 LoadPublicKey . . . 155
8.10.1.2.2.29 LoadSymmetricKey . . . 155
8.11 Header: ara/crypto/cryp/decryptor_private_ctx.h . . . 156
8.11.1 Class: DecryptorPrivateCtx . . . 156
8.11.1.1 Public Member Types . . . 157
8.11.1.1.1 Type Alias: Sptr . . . 157
8.11.1.2 Public Member Functions . . . 157
8.11.1.2.1 Member Functions . . . 157
8.11.1.2.1.1 Cancel . . . 157
8.11.1.2.1.2 Decrypt . . . 158
8.11.1.2.1.3 Finish . . . 159
8.11.1.2.1.4 GetMaximumChunkSize . . . 159
8.11.1.2.1.5 GetMinimumChunkSize . . . 160
8.11.1.2.1.6 GetRequiredOutputBufferSize . . . 160
8.11.1.2.1.7 SetKey . . . 161
8.11.1.2.1.8 SetKey . . . 162
8.11.1.2.1.9 Start . . . 162
8.11.1.2.1.10 Update . . . 163
8.12 Header: ara/crypto/cryp/encryptor_public_ctx.h . . . 164
8.12.1 Class: EncryptorPublicCtx . . . 164
8.12.1.1 Public Member Types . . . 164
8.12.1.1.1 Type Alias: Sptr . . . 164
8.12.1.2 Public Member Functions . . . 165
8.12.1.2.1 Member Functions . . . 165
8.12.1.2.1.1 Cancel . . . 165
8.12.1.2.1.2 Encrypt . . . 165
8.12.1.2.1.3 Finish . . . 166
8.12.1.2.1.4 GetMaximumChunkSize . . . 167
8.12.1.2.1.5 GetMinimumChunkSize . . . 167
8.12.1.2.1.6 GetRequiredOutputBufferSize . . . 168
8.12.1.2.1.7 SetKey . . . 169
8.12.1.2.1.8 SetKey . . . 169
8.12.1.2.1.9 Start . . . 170
8.12.1.2.1.10 Update . . . 171
8.13 Header: ara/crypto/cryp/hash_function_ctx.h . . . 171
8.13.1 Class: HashFunctionCtx . . . 171
8.13.1.1 Public Member Types . . . 172
8.13.1.1.1 Type Alias: Sptr . . . 172

8.13.1.2 Public Member Functions . . . 172
8.13.1.2.1 Member Functions . . . 172
8.13.1.2.1.1 CalculateHashDigest . . . 172
8.13.1.2.1.2 Cancel . . . 173
8.13.1.2.1.3 Compare . . . 174
8.13.1.2.1.4 ExportContextState . . . 174
8.13.1.2.1.5 Finish . . . 175
8.13.1.2.1.6 GetContextStateSize . . . 175
8.13.1.2.1.7 GetDigest . . . 176
8.13.1.2.1.8 GetDigestSize . . . 176
8.13.1.2.1.9 GetIntermediateDigest . . . 177
8.13.1.2.1.10 GetMaximumChunkSize . . . 178
8.13.1.2.1.11 ImportContextState . . . 178
8.13.1.2.1.12 Start . . . 179
8.13.1.2.1.13 Start . . . 179
8.13.1.2.1.14 Update . . . 180
8.13.1.2.1.15 Update . . . 181
8.14 Header: ara/crypto/cryp/key_agreement_private_ctx.h . . . 181
8.14.1 Class: KeyAgreementPrivateCtx . . . 181
8.14.1.1 Public Member Types . . . 182
8.14.1.1.1 Type Alias: Sptr . . . 182
8.14.1.2 Public Member Functions . . . 182
8.14.1.2.1 Member Functions . . . 182
8.14.1.2.1.1 AgreeKey . . . 182
8.14.1.2.1.2 SetKey . . . 183
8.14.1.2.1.3 SetKey . . . 184
8.14.1.2.1.4 SetParameter . . . 185
8.14.1.2.1.5 SetParameter . . . 185
8.15 Header: ara/crypto/cryp/key_decapsulator_private_ctx.h . . . 186
8.15.1 Class: KeyDecapsulatorPrivateCtx . . . 186
8.15.1.1 Public Member Types . . . 186
8.15.1.1.1 Type Alias: Sptr . . . 186
8.15.1.2 Public Member Functions . . . 187
8.15.1.2.1 Member Functions . . . 187
8.15.1.2.1.1 Decapsulate . . . 187
8.15.1.2.1.2 SetKey . . . 188
8.15.1.2.1.3 SetKey . . . 188
8.16 Header: ara/crypto/cryp/key_derivation_function_ctx.h . . . 189
8.16.1 Class: KeyDerivationFunctionCtx . . . 189
8.16.1.1 Public Member Types . . . 189
8.16.1.1.1 Type Alias: Sptr . . . 189
8.16.1.2 Public Member Functions . . . 190
8.16.1.2.1 Member Functions . . . 190

8.16.1.2.1.1 DeriveKey . . . 190
8.16.1.2.1.2 SetKey . . . 191
8.16.1.2.1.3 SetKey . . . 191
8.16.1.2.1.4 SetParameter . . . 192
8.16.1.2.1.5 SetParameter . . . 193
8.16.1.2.1.6 SetParameter . . . 193
8.17 Header: ara/crypto/cryp/key_encapsulator_public_ctx.h . . . 194
8.17.1 Class: KeyEncapsulatorPublicCtx . . . 194
8.17.1.1 Public Member Types . . . 194
8.17.1.1.1 Type Alias: Sptr . . . 194
8.17.1.2 Public Member Functions . . . 195
8.17.1.2.1 Member Functions . . . 195
8.17.1.2.1.1 Encapsulate . . . 195
8.17.1.2.1.2 Encapsulate . . . 196
8.17.1.2.1.3 GetRequiredOutputBufferSize . . . 197
8.17.1.2.1.4 SetKey . . . 198
8.17.1.2.1.5 SetKey . . . 198
8.18 Header: ara/crypto/cryp/keyslot.h . . . 199
8.18.1 Class: KeySlot . . . 199
8.18.1.1 Public Member Types . . . 199
8.18.1.1.1 Type Alias: Sptr . . . 199
8.18.1.2 Public Member Functions . . . 200
8.18.1.2.1 Special Member Functions . . . 200
8.18.1.2.1.1 Copy Constructor . . . 200
8.18.1.2.1.2 Move Constructor . . . 200
8.18.1.2.1.3 Move Assignment Operator . . . 201
8.18.1.2.1.4 Copy Assignment Operator . . . 201
8.18.1.2.1.5 Destructor . . . 201
8.18.1.2.2 Member Functions . . . 202
8.18.1.2.2.1 GetAlgId . . . 202
8.18.1.2.2.2 GetCryptoObjectType . . . 202
8.18.1.2.2.3 GetCryptoProvider . . . 203
8.18.1.2.2.4 GetPossibleAlgIds . . . 203
8.18.1.2.2.5 RegisterUpdateObserver . . . 204
8.18.1.2.2.6 UnregisterUpdateObserver . . . 204
8.19 Header: ara/crypto/cryp/message_authn_code_ctx.h . . . 205
8.19.1 Class: MessageAuthnCodeCtx . . . 205
8.19.1.1 Public Member Types . . . 205
8.19.1.1.1 Type Alias: Sptr . . . 205
8.19.1.2 Public Member Functions . . . 206
8.19.1.2.1 Member Functions . . . 206
8.19.1.2.1.1 Cancel . . . 206
8.19.1.2.1.2 Compare . . . 206

8.19.1.2.1.3 CompareSubspan . . . 207
8.19.1.2.1.4 Finish . . . 207
8.19.1.2.1.5 Generate . . . 208
8.19.1.2.1.6 GenerateSubspan . . . 209
8.19.1.2.1.7 GetDigest . . . 210
8.19.1.2.1.8 GetDigestSize . . . 210
8.19.1.2.1.9 GetMaximumChunkSize . . . 211
8.19.1.2.1.10 GetRecommendedIvSize . . . 211
8.19.1.2.1.11 SetKey . . . 212
8.19.1.2.1.12 SetKey . . . 212
8.19.1.2.1.13 Start . . . 213
8.19.1.2.1.14 Start . . . 214
8.19.1.2.1.15 Update . . . 214
8.19.1.2.1.16 Verify . . . 215
8.19.1.2.1.17 VerifySubspan . . . 216
8.20 Header: ara/crypto/cryp/protected_key_import_ctx.h . . . 217
8.20.1 Class: ProtectedKeyImportCtx . . . 217
8.20.1.1 Public Member Types . . . 217
8.20.1.1.1 Type Alias: Sptr . . . 217
8.20.1.2 Public Member Functions . . . 218
8.20.1.2.1 Member Functions . . . 218
8.20.1.2.1.1 GetCryptoObjectType . . . 218
8.20.1.2.1.2 GetPrivateKey . . . 218
8.20.1.2.1.3 GetPublicKey . . . 219
8.20.1.2.1.4 GetResponse . . . 220
8.20.1.2.1.5 GetSymmetricKey . . . 220
8.20.1.2.1.6 Import . . . 221
8.20.1.2.1.7 Reset . . . 222
8.20.1.2.1.8 SetKey . . . 222
8.20.1.2.1.9 SetKey . . . 223
8.21 Header: ara/crypto/cryp/random_generator_ctx.h . . . 224
8.21.1 Class: RandomGeneratorCtx . . . 224
8.21.1.1 Public Member Types . . . 224
8.21.1.1.1 Type Alias: Sptr . . . 224
8.21.1.2 Public Member Functions . . . 225
8.21.1.2.1 Member Functions . . . 225
8.21.1.2.1.1 AddEntropy . . . 225
8.21.1.2.1.2 Generate . . . 225
8.21.1.2.1.3 Seed . . . 226
8.21.1.2.1.4 Seed . . . 226
8.22 Header: ara/crypto/cryp/signer_private_ctx.h . . . 227
8.22.1 Class: SignerPrivateCtx . . . 227
8.22.1.1 Public Member Types . . . 228

8.22.1.1.1 Type Alias: Sptr . . . 228
8.22.1.2 Public Member Functions . . . 228
8.22.1.2.1 Member Functions . . . 228
8.22.1.2.1.1 Cancel . . . 228
8.22.1.2.1.2 Finish . . . 229
8.22.1.2.1.3 GetMaximumChunkSize . . . 229
8.22.1.2.1.4 GetSignatureSize . . . 230
8.22.1.2.1.5 SetKey . . . 230
8.22.1.2.1.6 SetKey . . . 231
8.22.1.2.1.7 Sign . . . 232
8.22.1.2.1.8 Start . . . 233
8.22.1.2.1.9 Start . . . 233
8.22.1.2.1.10 Update . . . 234
8.23 Header: ara/crypto/cryp/symmetric_cipher_ctx.h . . . 235
8.23.1 Class: SymmetricCipherCtx . . . 235
8.23.1.1 Public Member Types . . . 235
8.23.1.1.1 Type Alias: Sptr . . . 235
8.23.1.2 Public Member Functions . . . 236
8.23.1.2.1 Member Functions . . . 236
8.23.1.2.1.1 Cancel . . . 236
8.23.1.2.1.2 EncryptSingleCallWithOutputIv . . . 236
8.23.1.2.1.3 Finish . . . 237
8.23.1.2.1.4 FinishEncrypt . . . 238
8.23.1.2.1.5 GetMaximumChunkSize . . . 239
8.23.1.2.1.6 GetMaximumInputIvSize . . . 239
8.23.1.2.1.7 GetMinimumChunkSize . . . 240
8.23.1.2.1.8 GetMinimumInputIvSize . . . 240
8.23.1.2.1.9 GetOutputIvSize . . . 241
8.23.1.2.1.10 GetRequiredOutputBufferSize . . . 241
8.23.1.2.1.11 ProcessSingleCall . . . 242
8.23.1.2.1.12 ProcessSingleCallWithInputIv . . . 243
8.23.1.2.1.13 Seek . . . 244
8.23.1.2.1.14 SetKey . . . 245
8.23.1.2.1.15 SetKey . . . 246
8.23.1.2.1.16 Start . . . 246
8.23.1.2.1.17 Start . . . 247
8.23.1.2.1.18 Update . . . 248
8.24 Header: ara/crypto/cryp/symmetric_key_wrapper_ctx.h . . . 249
8.24.1 Class: SymmetricKeyWrapperCtx . . . 249
8.24.1.1 Public Member Types . . . 249
8.24.1.1.1 Type Alias: Sptr . . . 249
8.24.1.2 Public Member Functions . . . 250
8.24.1.2.1 Member Functions . . . 250

8.24.1.2.1.1 GetRequiredWrappedKeyBufferSize . . . 250
8.24.1.2.1.2 SetKey . . . 250
8.24.1.2.1.3 SetKey . . . 251
8.24.1.2.1.4 UnwrapSymmetricKey . . . 252
8.24.1.2.1.5 Wrap . . . 253
8.25 Header: ara/crypto/cryp/verifier_public_ctx.h . . . 254
8.25.1 Class: VerifierPublicCtx . . . 254
8.25.1.1 Public Member Types . . . 254
8.25.1.1.1 Type Alias: Sptr . . . 254
8.25.1.2 Public Member Functions . . . 255
8.25.1.2.1 Member Functions . . . 255
8.25.1.2.1.1 Cancel . . . 255
8.25.1.2.1.2 Finish . . . 255
8.25.1.2.1.3 GetMaximumChunkSize . . . 256
8.25.1.2.1.4 SetKey . . . 256
8.25.1.2.1.5 SetKey . . . 257
8.25.1.2.1.6 Start . . . 258
8.25.1.2.1.7 Start . . . 258
8.25.1.2.1.8 Update . . . 259
8.25.1.2.1.9 Verify . . . 260
8.25.1.2.1.10 Verify . . . 261
8.26 Header: ara/crypto/cryp/writeable_keyslot.h . . . 262
8.26.1 Class: WriteableKeySlot . . . 262
8.26.1.1 Public Member Types . . . 262
8.26.1.1.1 Type Alias: Sptr . . . 262
8.26.1.2 Public Member Functions . . . 263
8.26.1.2.1 Special Member Functions . . . 263
8.26.1.2.1.1 Move Constructor . . . 263
8.26.1.2.1.2 Copy Constructor . . . 263
8.26.1.2.1.3 Copy Assignment Operator . . . 264
8.26.1.2.1.4 Move Assignment Operator . . . 264
8.26.1.2.1.5 Destructor . . . 264
8.26.1.2.2 Member Functions . . . 265
8.26.1.2.2.1 Clear . . . 265
8.27 Header: ara/crypto/cryp/cryobj/crypto_object.h . . . 265
8.27.1 Class: CryptoObject . . . 265
8.27.1.1 Public Member Types . . . 266
8.27.1.1.1 Type Alias: Sptr . . . 266
8.27.1.2 Public Member Functions . . . 266
8.27.1.2.1 Member Functions . . . 266
8.27.1.2.1.1 GetAlgId . . . 266
8.27.1.2.1.2 GetAllowedUsage . . . 267
8.27.1.2.1.3 Save . . . 267

8.28 Header: ara/crypto/cryp/cryobj/private_key.h . . . 268
8.28.1 Class: PrivateKey . . . 268
8.28.1.1 Public Member Types . . . 268
8.28.1.1.1 Type Alias: Sptr . . . 268
8.28.1.2 Public Member Functions . . . 269
8.28.1.2.1 Member Functions . . . 269
8.28.1.2.1.1 GetPublicKey . . . 269
8.29 Header: ara/crypto/cryp/cryobj/public_key.h . . . 269
8.29.1 Class: PublicKey . . . 269
8.29.1.1 Public Member Types . . . 270
8.29.1.1.1 Type Alias: Sptr . . . 270
8.30 Header: ara/crypto/cryp/cryobj/symmetric_key.h . . . 270
8.30.1 Class: SymmetricKey . . . 270
8.30.1.1 Public Member Types . . . 271
8.30.1.1.1 Type Alias: Sptr . . . 271
8.31 Header: ara/crypto/x509/cert_sign_request.h . . . 271
8.31.1 Class: CertSignRequest . . . 271
8.31.1.1 Public Member Types . . . 272
8.31.1.1.1 Type Alias: Sptrc . . . 272
8.32 Header: ara/crypto/x509/certificate.h . . . 272
8.32.1 Non-Member Types . . . 272
8.32.1.1 Type Alias: X509ExtendedKeyUsage . . . 272
8.32.1.2 Type Alias: X509KeyUsage . . . 273
8.32.2 Global Variables . . . 273
8.32.2.1 kClientAuth . . . 273
8.32.2.2 kCodeSigning . . . 273
8.32.2.3 kConstrCrlSign . . . 274
8.32.2.4 kConstrDataEncipherment . . . 274
8.32.2.5 kConstrDecipherOnly . . . 275
8.32.2.6 kConstrDigitalSignature . . . 275
8.32.2.7 kConstrEncipherOnly . . . 276
8.32.2.8 kConstrKeyAgreement . . . 276
8.32.2.9 kConstrKeyCertSign . . . 277
8.32.2.10 kConstrKeyEncipherment . . . 277
8.32.2.11 kConstrNonRepudiation . . . 278
8.32.2.12 kEmailProtection . . . 278
8.32.2.13 kOCSPSigning . . . 279
8.32.2.14 kServerAuth . . . 279
8.32.2.15 kTimeStamping . . . 279
8.32.3 Class: Certificate . . . 280
8.32.3.1 Public Member Types . . . 280
8.32.3.1.1 Type Alias: Sptrc . . . 280
8.32.3.2 Public Member Functions . . . 281

8.32.3.2.1 Member Functions . . . 281
8.32.3.2.1.1 GetAuthorityInformationAccessLocation . . . 281
8.32.3.2.1.2 GetAuthorityKeyIdentifierExtension . . . 281
8.32.3.2.1.3 GetCertificateSignatureAlgorithm . . . 282
8.32.3.2.1.4 GetCrlDistributionPoints . . . 282
8.32.3.2.1.5 GetExtendedKeyUsageExtension . . . 283
8.32.3.2.1.6 GetIssuer . . . 283
8.32.3.2.1.7 GetKeyUsageExtension . . . 284
8.32.3.2.1.8 GetNotAfter . . . 284
8.32.3.2.1.9 GetNotBefore . . . 285
8.32.3.2.1.10 GetSerialNumber . . . 285
8.32.3.2.1.11 GetSubject . . . 286
8.32.3.2.1.12 GetSubjectKeyIdentifierExtension . . . 286
8.32.3.2.1.13 GetSubjectPublicKeyAlgorithm . . . 287
8.32.3.2.1.14 LoadSubjectPublicKey . . . 287
8.33 Header: ara/crypto/x509/certificate_validation_ctx.h . . . 288
8.33.1 Non-Member Types . . . 288
8.33.1.1 Type Alias: CertificateValidationOptions . . . 288
8.33.2 Global Variables . . . 289
8.33.2.1 kIgnoreCrl . . . 289
8.33.2.2 kIgnoreNotAfter . . . 289
8.33.2.3 kIgnoreNotBefore . . . 290
8.33.2.4 kIgnoreUnknownCriticalExtensions . . . 290
8.33.2.5 kOcspOff . . . 291
8.33.2.6 kOcspResponseCompleteChain . . . 291
8.33.2.7 kOcspResponseTargetOnly . . . 292
8.33.3 Class: CertificateValidationCtx . . . 292
8.33.3.1 Public Member Types . . . 293
8.33.3.1.1 Type Alias: Sptr . . . 293
8.33.3.2 Public Member Functions . . . 293
8.33.3.2.1 Member Functions . . . 293
8.33.3.2.1.1 AddOcspResponse . . . 293
8.33.3.2.1.2 AddTrustedCertificate . . . 294
8.33.3.2.1.3 AddUntrustedCertificate . . . 294
8.33.3.2.1.4 IgnoreCriticalExtension . . . 295
8.33.3.2.1.5 SetCurrentTime . . . 296
8.33.3.2.1.6 SetValidationOptions . . . 296
8.33.3.2.1.7 ValidateCertificate . . . 297
8.33.3.2.1.8 ValidateCertificateChain . . . 297
8.34 Header: ara/crypto/x509/certificate_validation_result.h . . . 298
8.34.1 Non-Member Types . . . 298
8.34.1.1 Enumeration: CertificateValidationStatus . . . 298
8.34.2 Class: CertificateValidationResult . . . 299

8.34.2.1 Public Member Types . . . 300
8.34.2.1.1 Type Alias: Sptrc . . . 300
8.34.2.2 Public Member Functions . . . 300
8.34.2.2.1 Member Functions . . . 300
8.34.2.2.1.1 GetCertificateChain . . . 300
8.34.2.2.1.2 GetFailedCertificate . . . 301
8.34.2.2.1.3 GetValidationStatus . . . 301
8.35 Header: ara/crypto/x509/ocsp_response.h . . . 302
8.35.1 Non-Member Types . . . 302
8.35.1.1 Enumeration: OcspResponseStatus . . . 302
8.35.2 Class: OcspResponse . . . 302
8.35.2.1 Public Member Types . . . 303
8.35.2.1.1 Type Alias: Sptrc . . . 303
8.35.2.2 Public Member Functions . . . 303
8.35.2.2.1 Member Functions . . . 303
8.35.2.2.1.1 GetCertificates . . . 303
8.35.2.2.1.2 GetResponseStatus . . . 304
8.36 Header: ara/crypto/x509/x509_dn.h . . . 304
8.36.1 Class: X509DN . . . 304
8.36.1.1 Public Member Types . . . 305
8.36.1.1.1 Type Alias: Sptrc . . . 305
8.36.1.2 Public Member Functions . . . 305
8.36.1.2.1 Member Functions . . . 305
8.36.1.2.1.1 ToString . . . 305
8.37 Header: ara/crypto/x509/x509_provider.h . . . 306
8.37.1 Class: X509Provider . . . 306
8.37.1.1 Public Member Types . . . 306
8.37.1.1.1 Type Alias: Sptr . . . 306
8.37.1.2 Public Member Functions . . . 307
8.37.1.2.1 Special Member Functions . . . 307
8.37.1.2.1.1 Move Constructor . . . 307
8.37.1.2.1.2 Copy Constructor . . . 307
8.37.1.2.1.3 Move Assignment Operator . . . 308
8.37.1.2.1.4 Copy Assignment Operator . . . 308
8.37.1.2.1.5 Destructor . . . 308
8.37.1.2.2 Member Functions . . . 309
8.37.1.2.2.1 ClearCertificateSlot . . . 309
8.37.1.2.2.2 CreateCertSignRequest . . . 309
8.37.1.2.2.3 CreateCertificateValidationCtx . . . 310
8.37.1.2.2.4 CreateDn . . . 311
8.37.1.2.2.5 CreateDn . . . 311
8.37.1.2.2.6 CreateOcspRequest . . . 312
8.37.1.2.2.7 CreateOcspRequest . . . 313

8.37.1.2.2.8 DeleteCrl . . . 313
8.37.1.2.2.9 ImportCertificate . . . 314
8.37.1.2.2.10 ImportCertificateInGroup . . . 315
8.37.1.2.2.11 ImportCrl . . . 315
8.37.1.2.2.12 ParseCert . . . 316
8.37.1.2.2.13 ParseCertChain . . . 317
8.37.1.2.2.14 ParseOcspResponse . . . 318
8.37.1.2.2.15 RemoveCertificateFromGroup . . . 318
8.37.1.2.3 Named Constructors . . . 319
8.37.1.2.3.1 LoadCertificate . . . 319
8.37.1.2.3.2 LoadCertificateGroup . . . 320
9 Service Interfaces 321
10 Configuration 322
10.1 Default Values . . . 322
10.2 Semantic Constraints . . . 322
A Mentioned Manifest Elements 323
B Interfaces to other Functional Clusters (informative) 331
B.1 Overview . . . 331
B.2 Interface Tables . . . 331
C Change History 332
C.1 Change History of this document according to AUTOSAR Release R22-11 332
C.1.1 Added Specification Items in R22-11 . . . 332
C.1.2 Changed Specification Items in R22-11 . . . 337
C.1.3 Deleted Specification Items in R22-11 . . . 337
C.2 Change History of this document according to AUTOSAR Release R23-11 338
C.2.1 Added Specification Items in R23-11 . . . 338
C.2.2 Changed Specification Items in R23-11 . . . 338
C.2.3 Deleted Specification Items in R23-11 . . . 339
C.3 Change History of this document according to AUTOSAR Release R24-11 339
C.3.1 Added Specification Items in R24-11 . . . 339
C.3.2 Changed Specification Items in R24-11 . . . 339
C.3.3 Deleted Specification Items in R24-11 . . . 342
C.3.4 Added Constraints in R24-11 . . . 343
C.3.5 Changed Constraints in R24-11 . . . 343
C.3.6 Deleted Constraints in R24-11 . . . 343
C.4 Change History of this document according to AUTOSAR Release R25-11 343
C.4.1 Added Specification Items in R25-11 . . . 343
C.4.2 Changed Specification Items in R25-11 . . . 345
C.4.3 Deleted Specification Items in R25-11 . . . 346
C.4.4 Added Constraints in R25-11 . . . 350
C.4.5 Changed Constraints in R25-11 . . . 350

C.4.6 Deleted Constraints in R25-11 350
D Naming Convention 351

1 Introduction and functional overview

This specification describes the functionality and the configuration for the Adaptive AUTOSAR Functional Cluster Cryptography (FC Crypto) and its API (CryptoAPI, which is part of the AUTOSAR Adaptive Platform Foundation.

The FC Crypto offers applications and other Adaptive AUTOSAR Functional Clusters a standardized interface for operations including cryptographic operations, key management, and certificate handling. FC Crypto manages the actual implementations of all operations, the configuration, and the brokering of operations from applications to implementations. The standardized interface is exposed by the CryptoAPI.

2 Acronyms and Abbreviations

The glossary below includes acronyms and abbreviations relevant to the FC Crypto module that are not included in the [1, AUTOSAR glossary].

Abbreviation / Acronym:	Description:
ACL	Access Control List
AE	Authenticated Encryption
AEAD	Authenticated Encryption with Associated Data – Encryption scheme which simultaneously provides confidentiality and authenticity of data as well as additional authenticated but not encrypted data.
AES	Advanced Encryption Standard – A block cipher for the symmetric encryption of electronic data.
API	Abstract Programming Interface
ARA	Autosar Runtime Environment for Adaptive Applications
ASN.1	Abstract Syntax Notation One, as defined in the ASN.1 standards
BER	Basic Encoding Rules
BLOB	Binary Large Object – A Binary Large OBject (BLOB) is a collection of binary data stored as a single entity.
CA	Certificate Authority or Certification Authority is an entity that issues digital certificates.
CBC	Cipher Block Chaining Mode – A mode of operation for symmetric ciphers (e.g. AES) that supports encryption.
CBC-MAC	Cipher Block Chaining Message Authentication Mode – A mode of operation for symmetric ciphers (e.g. AES) that supports authentication.
CCM	Counter Mode with CBC-MAC – An AEAD operation mode (encryption and authentication) for AES.
CMAC	Cipher-based Message Authentication Code – A mode of operation for symmetric ciphers (e.g. AES) that supports authentication and is similar but advanced to CBC-MAC.
CMP	X.509 Certificate Management Provider.
CO	Cryptographic Object
COUID	Cryptographic Object Unique Identifier
CRL	Certificate Revocation Lists is a list of digital certificates that have been revoked before their expiration date was reached. This list contains all the serial numbers of the revoked certificates and the revoked data.
CSR	Certificate Signing Request
CTL	Certificate Trust List is a list of digital certificates that are explicitly trusted in this environment. This list contains all the serial numbers of the explicitly trusted certificates.
DER	Distinguished Encoding Rules as defined in [2]
DH	Diffie-Hellman (key exchange method)
ECC	Elliptic Curve Cryptography – Public-key cryptography based on the structure of elliptic curves.
ECDH	Elliptic Curve Diffie-Hellman – An ECC based DH key exchange with perfect forward secrecy.
ECDSA	Elliptic Curve Digital Signature Algorithm – An ECC based signature scheme.
ECIES	Elliptic Curve Integrated Encryption Scheme – An ECC based encryption scheme.

Abbreviation / Acronym:	Description:
ECU	Electronic Control Unit
FC Crypto	Functional cluster Cryptography. This is the AUTOSAR cluster, which provides all important functionality related to cryptograhic, key management, and certificate handling needs.
gamma	linear recurrent sequence
GCM	Galois Counter Mode – An AEAD operation mode (encryption and authentication) for AES.
GMAC	Galois MAC – A mode of operation for symmetric ciphers (e.g. AES) that supports authentication.
HSM	Hardware Security Module – Hardware security module, used to store cryptographic credentials and secure run-time environment
HMAC	Hashed Message Authentication Code
IAM	Identity and Access Management
IETF	Internet Engineering Task Force
IKE	Internet Key Exchange
IPC	Inter-Process Communication
IPsec	Internet Protocol Security (IPsec) is a secure network protocol suite that authenticates and encrypts the packets of data to provide secure encrypted communication between two computers over an Internet Protocol network.
IV	Initialization Vector
KDF	Key Derivation Function – A function to derive one or more keys from a secret value.
KEK	Key encryption key – A key that is used to encrypt another key for transportation or storage in an unsecure environment
KEM	Key Encapsulation Mechanism
KSP	Key Storage Provider
MAC	Message Authentication Code
MGF	Mask Generation Function – A cryptographic function similar to a hash function. It takes a variable length input and an output length l to generate an output of length l. If the input is unknown, the output appears random.
OCSP	Online Certificate Status Protocol – Internet protocol used to obtain revocation status of X.509 certificates.
PDP	Policy Decision Point
PEP	Policy Enforcement Point
PEM	Privacy-Enhanced Mail
PKI	Public Key Infrastructure – A system that issues, distributes, and checks digital certificates.
PKCS	Public Key Cryptography Standard.
RA	Registration Authority
RNG	Random Number Generator
RSA	Rivest, Shamir, Adleman – RSA is an algorithm for public-key cryptography; It is named after its inventors Ronald L. Rivest, Adi Shamir and Leonard Adleman.
SecOC	Secure Onboard Communication
SHA-1	Secure Hash Algorithm (version 1) – Hash functions family.
SHA-2	Secure Hash Algorithm (version 2) – Hash functions family with different hash value length.
SHA-3	Secure Hash Algorithm (version 3) – New hash function generation, faster and more secure as SHA-2.
SHE	Secure Hardware Extension

Abbreviation / Acronym:	Description:
TLS	Transport Layer Security (TLS) is a cryptographic protocol designed to provide communications security over a computer network.
TPM	The Trusted Platform Module is defined in [3] and is a secure cryptoprocessor.
UCM	Update and Configuration Management
UID	Unique Identifier
X.509	Standard for certificates
IDSM	Intrusion Detection System Manager
SEV	Security Event

Terms:	Description:
Adaptive Application	An adaptive application is a part of application SW in the architecture of Adaptive AUTOSAR. An adaptive application runs on top of ARA and accesses AUTOSAR functional clusters through ARA.
Adaptive Platform Services	Adaptive Platform Services are located below the ARA. They provide platform standard services of Adaptive AUTOSAR.
Asymmetric Key	An asymmetric key describes a pair of two keys (public and private key). A cipher text created by one key cannot be decrypted with this key. Encryption is only possible with the other key of this pair.
Block Cipher	A symmetric encryption that encrypts plaintext blocks of fixed length.
certificate serial number	An integer value, unique within the issuing authority, which is unambiguously associated with a certificate issued by that authority.
Certificate Slot	Secure storage of certificate material. Certificate slots define the access to the stored certificate material and may grant the access only to authorized application or functional cluster.
certification path	An ordered list of one or more public-key certificates, starting with a public-key certificate signed by the trust anchor, and ending with the public key certificate to be validated. All intermediate public-key certificates, if any, are CA-certificates in which the subject of the preceding certificate is the issuer of the following certificate.
Ciphertext	A ciphertext is an encrypted text, which is the result of encryption performed on `plaintext`.
CryptoAPI	The set of all interfaces that are provided by FC Crypto to consumers.
Crypto Provider	A structural element that organizes cryptographic primitives.
Cryptographic primitives	Well-established, low-level cryptographic algorithms that are frequently used to build cryptographic protocols for computer security systems.
Distinguished name	is originally defined in X.501 [4] as a representation of a directory name, defined as a construct that identifies a particular object from among a set of all objects.
Functional Cluster	The SW functionality of ARA is divided into functional clusters. Functional clusters provide APIs and can communicate with each other.
Identity and Access Management	A functional cluster of adaptive AUTOSAR.

Terms:	Description:
Instance Specifier	Crypto provider can have more than one instance. To distinguish between instances the spcific instance is addressed with an instance specifier. An instance specifier identifies one instance of a crypto provider.
Key Material	public keys, private keys, seeds.
Key Slot	Secure storage of key material. Key slots define the access to the stored key material and grant the access only to authorized application or functional cluster.
Key Storage Provider	A structural element that organizes and manages cryptographic keys.
Message Authentication Code	A cryptographic function similar to a hash function. It takes a message of variable length and a secret key as input to generate a hash value, the MAC value. The MAC value is attached to the message to be sent. The receiver of the message can recalculate the MAC value to check if the message is authentic.
Nonce	A nonce is a random or semi-random number that is generated for cryptographic topics. A nonce can be used as an input to a hash algorithm so that the hash algorithm computes a hash value out of two inputs: plaintext and nonce. Usage of nonces enhances security against brute force attacks.
Plaintext	A plaintext is ordinary readable text before being encrypted into ciphertext or after being decrypted.
Policy Decision Point	A PDP defines which item (process, application, function) can decide if a requested access to resources may be granted or not.
Policy Enforcement Point	A PEP is the point a policy decision is used to grant or deny the access.
Random Number Generator	A program that generates random numbers or pseudo random numbers in a given range.
Salt	A salt is a random or semi-random number which is created for passwords. When a password is edited for a user/account also a salt is created for this user/account. A hash algorithm creates a hash value of password and salt. Salts increase the security against brute force password guessing attacks.
SecretSeed	A secret value that is used as an initial value to start encryption/decryption.
Stream Cipher	A symmetric encryption that calculates cipher text out of streaming plaintext and the status result of the encryption of previous streamed plaintext. For the first part of encryption a start value is needed as status result.
Symmetric Key	In a symmetric encryption the same key (symmetric key) is used to encrypt plaintext into cipher text and to decode cipher text into plaintext. A symmetric key is also called secret key because it must be kept secret.
X.509 Provider	Domain SW for X.509 certificates parsing, verification, storage and search.

3 Related documentation

3.1 Input documents & related standards and norms

[1] Glossary
AUTOSAR_FO_TR_Glossary

[2] X.690 : Information technology - ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)
https://www.itu.int/rec/T-REC-X.690

[3] ISO 11889 – ISO/IEC 11889-1:2015 Information technology - Trusted platform module library - Part 1: Architecture
https://www.iso.org

[4] X.501 : Information technology - Open Systems Interconnection - The Directory: Models
https://www.itu.int/rec/T-REC-X.501

[5] Specification of Adaptive Platform Core
AUTOSAR_AP_SWS_Core

[6] Explanation of Adaptive Platform Software Architecture
AUTOSAR_AP_EXP_SWArchitecture

[7] General Requirements specific to Adaptive Platform
AUTOSAR_AP_RS_General

[8] Requirements on Cryptography
AUTOSAR_AP_RS_Cryptography

[9] BSI: Functionality Classes and Evaluation Methodology for Deterministic Random Number Generators (AIS)
https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Zertifizierung/Interpretationen/AIS_20_Functionality_Classes_Evaluation_Methodology_DRNG_e.pdf?__blob=publicationFile[5]

[10] Using Advanced Encryption Standard Counter Mode (AES-CTR) with the Internet Key Exchange version 02 (IKEv2) Protocol
https://rfc-editor.org/rfc/rfc5930.txt

[11] ChaCha20-Poly1305 Cipher Suites for Transport Layer Security (TLS)
https://rfc-editor.org/rfc/rfc7905.txt

[12] Advanced Encryption Standard (AES) Key Wrap Algorithm
https://tools.ietf.org/html/rfc3394

[13] Advanced Encryption Standard (AES) Key Wrap with Padding Algorithm
https://tools.ietf.org/html/rfc5649

[14] New directions in cryptography

https://ieeexplore.ieee.org/document/1055638

[15] X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP
https://rfc-editor.org/rfc/rfc6960.txt

[16] Specification of Manifest
AUTOSAR_AP_TPS_ManifestSpecification

[17] Specification of Execution Management
AUTOSAR_AP_SWS_ExecutionManagement

3.2 Further applicable specification

AUTOSAR provides a core specification [5, SWS AdaptivePlatformCore] which is also applicable for FC Crypto. The chapter "General requirements for all FunctionalClusters" of this specification shall be considered as an additional and required specification for implementation of FC Crypto.

4 Constraints and assumptions

4.1 Assumptions on the Implementation of Crypto

4.1.1 Process Isolation

In order to perform access control on keys and certificates, the implementation of Crypto must not be limited to the application process, i.e., it is assumed that it is not executed as a library-only implementation. Instead, the implementation of Crypto may relay on separate processes or other forms of trust boundaries (e.g., trusted execution environments).

4.1.2 Usage of Cryptographic Modules

It is assumed that implementations of Crypto may make use of cryptographic module implementations that are implementated by a different party than the stack vendor and are exposed via a standardized interface (e.g., PSA, PKCS#11).

4.2 Known limitations

There are no known limitations for this specification.

5 Dependencies to other functional clusters

This chapter provides an overview of the dependencies to other Functional Clusters in the AUTOSAR Adaptive Platform. Section 5.1 “Provided Interfaces” lists the interfaces provided by `Cryptography` to other Functional Clusters. Section 5.2 “Required Interfaces” lists the interfaces required by `Cryptography`.

A detailed technical architecture documentation of the AUTOSAR Adaptive Platform is provided in [6].

5.1 Provided Interfaces

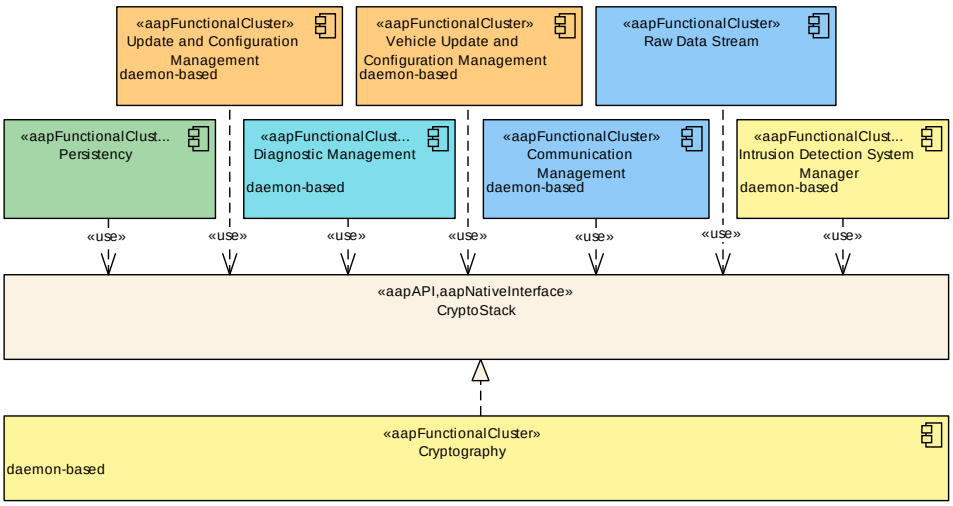

Figure 5.1: Interfaces provided by Cryptography to other Functional Clusters

Figure 5.1 shows interfaces assumed to be provided by `Cryptography` to other Functional Clusters within the AUTOSAR Adaptive Platform. Table 5.1 provides a list of interfaces assumed to be provided to other Functional Clusters within the AUTOSAR Adaptive Platform. Please note that for brevity only a placeholder interface `CryptoStack` is used instead of the many different standardized interfaces that would be required e.g., to access a certain key or to calculate a checksum.

Interface	*Functional Cluster*	*Purpose*
`CryptoStack`	Communication Management	This interface may be used e.g., to establish encrypted connections and generate / verify checksums (MAC).
	Diagnostic Management	This interface may be used e.g., to access keys for secure diagnostics.
	Intrusion Detection System Manager	`Adaptive Intrusion Detection System Manager` uses this interface to sign security events.
	Persistency	Used to ensure confidentiality and integrity of the persisted data.
	Raw Data Stream	This interface may be used to establish encrypted connections.

△

Interface	*Functional Cluster*	*Purpose*
	Time Synchronization	`Time Synchronization` shall use this interface to generate / verify checksums (MAC).
	Update and Configuration Management	This interface may be used e.g., to verify the integrity and authenticity of `Software Package`s.
	Vehicle Update and Configuration Management	This interface may be used e.g., to verify the integrity and authenticity of `Vehicle Package`s.

Table 5.1: Interfaces provided to other Functional Clusters

5.2 Required Interfaces

Table 5.2 provides a list of interfaces assumed to be required from other Functional Clusters within the AUTOSAR Adaptive Platform.

Functional Cluster	*Interface*	*Purpose*
Execution Management	ExecutionClient	

Table 5.2: Interfaces required from other Functional Clusters

6 Requirements Tracing

The following tables reference the requirements specified in [7] and [8] and links to the fulfillment of these. Please note that if column “Satisfied by” is empty for a specific requirement this means that this requirement is not fulfilled by this document.

Requirement	Description	Satisfied by
[RS_AP_00130]	AUTOSAR Adaptive Platform shall represent a rich and modern programming environment	[SWS_CRYPT_19900]
[RS_AP_00144]	Availability of a named constructor	[SWS_CRYPT_20745] [SWS_CRYPT_20746] [SWS_CRYPT_20747] [SWS_CRYPT_20748] [SWS_CRYPT_20750] [SWS_CRYPT_20751] [SWS_CRYPT_20752] [SWS_CRYPT_20753] [SWS_CRYPT_20756] [SWS_CRYPT_20757] [SWS_CRYPT_20758]
[RS_AP_00158]	IAM access violations	[SWS_CRYPT_41056]
[RS_CRYPTO_02001]	The Crypto Stack shall conceal symmetric keys from the users	[SWS_CRYPT_00007] [SWS_CRYPT_20762] [SWS_CRYPT_20763] [SWS_CRYPT_20764] [SWS_CRYPT_21313] [SWS_CRYPT_21413] [SWS_CRYPT_21525] [SWS_CRYPT_21815] [SWS_CRYPT_22118] [SWS_CRYPT_23515] [SWS_CRYPT_23800] [SWS_CRYPT_23911] [SWS_CRYPT_24018] [SWS_CRYPT_24115] [SWS_CRYPT_50010] [SWS_CRYPT_50012] [SWS_CRYPT_50018] [SWS_CRYPT_50026] [SWS_CRYPT_50034] [SWS_CRYPT_50036] [SWS_CRYPT_50043] [SWS_CRYPT_50053] [SWS_CRYPT_50066] [SWS_CRYPT_50072] [SWS_CRYPT_50083] [SWS_CRYPT_50086] [SWS_CRYPT_50138] [SWS_CRYPT_50157] [SWS_CRYPT_50175]
[RS_CRYPTO_02002]	The Crypto Stack shall conceal asymmetric private keys from the users	[SWS_CRYPT_00007] [SWS_CRYPT_20762] [SWS_CRYPT_20763] [SWS_CRYPT_20764] [SWS_CRYPT_22500]
[RS_CRYPTO_02003]	The Crypto Stack shall support management of non-persistent session/ephemeral keys during their lifetime	[SWS_CRYPT_20721] [SWS_CRYPT_20722] [SWS_CRYPT_20955] [SWS_CRYPT_21313] [SWS_CRYPT_21413] [SWS_CRYPT_21525] [SWS_CRYPT_21815] [SWS_CRYPT_22118] [SWS_CRYPT_23515] [SWS_CRYPT_23911] [SWS_CRYPT_24018] [SWS_CRYPT_24115] [SWS_CRYPT_50010] [SWS_CRYPT_50012] [SWS_CRYPT_50018] [SWS_CRYPT_50026] [SWS_CRYPT_50034] [SWS_CRYPT_50036] [SWS_CRYPT_50043] [SWS_CRYPT_50053] [SWS_CRYPT_50066] [SWS_CRYPT_50072] [SWS_CRYPT_50083] [SWS_CRYPT_50086] [SWS_CRYPT_50138] [SWS_CRYPT_50157] [SWS_CRYPT_50175]

△

Requirement	Description	Satisfied by
[RS_CRYPTO_02004]	The Crypto Stack shall support secure storage of cryptographic artifacts	[SWS_CRYPT_00102] [SWS_CRYPT_00103] [SWS_CRYPT_04203] [SWS_CRYPT_04205] [SWS_CRYPT_04210] [SWS_CRYPT_04211] [SWS_CRYPT_10000] [SWS_CRYPT_10001] [SWS_CRYPT_10016] [SWS_CRYPT_10019] [SWS_CRYPT_10031] [SWS_CRYPT_10033] [SWS_CRYPT_10701] [SWS_CRYPT_10710] [SWS_CRYPT_20517] [SWS_CRYPT_30204] [SWS_CRYPT_30205] [SWS_CRYPT_30214] [SWS_CRYPT_30215] [SWS_CRYPT_30216] [SWS_CRYPT_30217] [SWS_CRYPT_30226] [SWS_CRYPT_30227] [SWS_CRYPT_40997] [SWS_CRYPT_40998] [SWS_CRYPT_41003] [SWS_CRYPT_41004] [SWS_CRYPT_41005] [SWS_CRYPT_41006] [SWS_CRYPT_41013] [SWS_CRYPT_41014] [SWS_CRYPT_41021] [SWS_CRYPT_41023] [SWS_CRYPT_50004] [SWS_CRYPT_50005] [SWS_CRYPT_50147] [SWS_CRYPT_50148] [SWS_CRYPT_50149] [SWS_CRYPT_50150] [SWS_CRYPT_50168] [SWS_CRYPT_50169] [SWS_CRYPT_50170] [SWS_CRYPT_50171] [SWS_CRYPT_50192] [SWS_CRYPT_50193] [SWS_CRYPT_50194] [SWS_CRYPT_50195] [SWS_CRYPT_50196] [SWS_CRYPT_60000] [SWS_CRYPT_60001] [SWS_CRYPT_60002] [SWS_CRYPT_60003] [SWS_CRYPT_60005] [SWS_CRYPT_60006] [SWS_CRYPT_60012] [SWS_CRYPT_60013] [SWS_CRYPT_60014] [SWS_CRYPT_60015] [SWS_CRYPT_60016] [SWS_CRYPT_60017] [SWS_CRYPT_60018] [SWS_CRYPT_60019] [SWS_CRYPT_60021] [SWS_CRYPT_60022]
[RS_CRYPTO_02005]	The Crypto Stack shall support unique identification of cryptographic objects	[SWS_CRYPT_20500] [SWS_CRYPT_20501] [SWS_CRYPT_20505] [SWS_CRYPT_50092]
[RS_CRYPTO_02007]	The Crypto Stack shall provide means for secure handling of "secret seeds"	[SWS_CRYPT_00102]
[RS_CRYPTO_02008]	The Crypto Stack shall support restrictions of the allowed usage scope for keys and "secret seeds"	[SWS_CRYPT_20400] [SWS_CRYPT_20401] [SWS_CRYPT_20411]
[RS_CRYPTO_02009]	The Crypto stack shall support separation of applications" access rights for each cryptographic object slot	[SWS_CRYPT_30405] [SWS_CRYPT_50054] [SWS_CRYPT_50055] [SWS_CRYPT_50056]
[RS_CRYPTO_02101]	The Crypto Stack shall provide interfaces to generate cryptographic keys for all supported primitives	[SWS_CRYPT_20721] [SWS_CRYPT_20722] [SWS_CRYPT_40962]
[RS_CRYPTO_02102]	The Crypto Stack shall prevent keys from being used in incompatible or insecure ways	[SWS_CRYPT_00102] [SWS_CRYPT_10014] [SWS_CRYPT_20721] [SWS_CRYPT_20722] [SWS_CRYPT_21412] [SWS_CRYPT_21515] [SWS_CRYPT_21813] [SWS_CRYPT_41052] [SWS_CRYPT_50000] [SWS_CRYPT_50007] [SWS_CRYPT_50008] [SWS_CRYPT_50009] [SWS_CRYPT_50011] [SWS_CRYPT_50051] [SWS_CRYPT_50052]
[RS_CRYPTO_02103]	The Crypto Stack shall support primitives to derive cryptographic key material from a base key material	[SWS_CRYPT_20748] [SWS_CRYPT_21500] [SWS_CRYPT_21501]

▽

△

Requirement	Description	Satisfied by
[RS_CRYPTO_02104]	The Crypto Stack shall support a primitive to exchange cryptographic keys with another entity	[SWS_CRYPT_20743] [SWS_CRYPT_20752] [SWS_CRYPT_20753] [SWS_CRYPT_20758] [SWS_CRYPT_21300] [SWS_CRYPT_21301] [SWS_CRYPT_21400] [SWS_CRYPT_21401] [SWS_CRYPT_21800] [SWS_CRYPT_24000]
[RS_CRYPTO_02105]	Symmetric keys and asymmetric private keys shall be imported and exported in a secure format.	[SWS_CRYPT_10700] [SWS_CRYPT_20728] [SWS_CRYPT_20732] [SWS_CRYPT_41020] [SWS_CRYPT_50014] [SWS_CRYPT_50017] [SWS_CRYPT_50084] [SWS_CRYPT_50085] [SWS_CRYPT_50132] [SWS_CRYPT_50190] [SWS_CRYPT_50205] [SWS_CRYPT_50206] [SWS_CRYPT_50207] [SWS_CRYPT_50208] [SWS_CRYPT_50209] [SWS_CRYPT_50210] [SWS_CRYPT_50211] [SWS_CRYPT_50212]
[RS_CRYPTO_02107]	The Crypto Stack shall support the algorithm specification in any key generation or derivation request	[SWS_CRYPT_10014] [SWS_CRYPT_20710] [SWS_CRYPT_20721] [SWS_CRYPT_20722] [SWS_CRYPT_21515] [SWS_CRYPT_50000] [SWS_CRYPT_50007] [SWS_CRYPT_50008] [SWS_CRYPT_50009] [SWS_CRYPT_50051] [SWS_CRYPT_50052]
[RS_CRYPTO_02108]	The Crypto Stack shall provide interfaces for management and usage of algorithm-specific domain parameters	[SWS_CRYPT_20721] [SWS_CRYPT_20722] [SWS_CRYPT_21412] [SWS_CRYPT_21515] [SWS_CRYPT_21813] [SWS_CRYPT_22511] [SWS_CRYPT_50007] [SWS_CRYPT_50008] [SWS_CRYPT_50009] [SWS_CRYPT_50011] [SWS_CRYPT_50051] [SWS_CRYPT_50052]
[RS_CRYPTO_02109]	The Crypto Stack shall support interfaces for a unified Machine-wide storage and retrieval of different crypto objects	[SWS_CRYPT_20701]
[RS_CRYPTO_02110]	The Crypto Stack shall support prototyping of application-exclusive key slot resources	[SWS_CRYPT_00101] [SWS_CRYPT_41043] [SWS_CRYPT_41044] [SWS_CRYPT_41045] [SWS_CRYPT_41047] [SWS_CRYPT_41049] [SWS_CRYPT_41051] [SWS_CRYPT_41056]
[RS_CRYPTO_02111]	The Crypto Stack shall provide applications a possibility to define usage restrictions of any new generated or derived key	[SWS_CRYPT_10015] [SWS_CRYPT_13101] [SWS_CRYPT_13102] [SWS_CRYPT_13103] [SWS_CRYPT_13104] [SWS_CRYPT_13105] [SWS_CRYPT_13106] [SWS_CRYPT_13107] [SWS_CRYPT_13108] [SWS_CRYPT_13109] [SWS_CRYPT_13110] [SWS_CRYPT_13150] [SWS_CRYPT_20721] [SWS_CRYPT_20722] [SWS_CRYPT_21515] [SWS_CRYPT_40991] [SWS_CRYPT_40992] [SWS_CRYPT_50001] [SWS_CRYPT_50002] [SWS_CRYPT_50007] [SWS_CRYPT_50008] [SWS_CRYPT_50009] [SWS_CRYPT_50051] [SWS_CRYPT_50052]
[RS_CRYPTO_02112]	The Crypto Stack shall execute export/import of a key value together with its meta information	[SWS_CRYPT_10711] [SWS_CRYPT_20728] [SWS_CRYPT_20732] [SWS_CRYPT_50006] [SWS_CRYPT_50014] [SWS_CRYPT_50132] [SWS_CRYPT_50197] [SWS_CRYPT_50198] [SWS_CRYPT_50199] [SWS_CRYPT_50200]
[RS_CRYPTO_02115]	The Crypto Stack shall enforce assigning required domain parameters to a key in its generation or derivation procedure	[SWS_CRYPT_20721] [SWS_CRYPT_20722] [SWS_CRYPT_21312] [SWS_CRYPT_21412] [SWS_CRYPT_21515] [SWS_CRYPT_21813] [SWS_CRYPT_22511] [SWS_CRYPT_24016] [SWS_CRYPT_50011]

▽

△

Requirement	Description	Satisfied by
[RS_CRYPTO_02201]	The Crypto Stack shall provide interfaces to use symmetric encryption and decryption primitives	[SWS_CRYPT_20742] [SWS_CRYPT_23801] [SWS_CRYPT_24001] [SWS_CRYPT_24014] [SWS_CRYPT_50136] [SWS_CRYPT_50137] [SWS_CRYPT_50155] [SWS_CRYPT_50156] [SWS_CRYPT_50173] [SWS_CRYPT_50174]
[RS_CRYPTO_02202]	The Crypto Stack shall provide interfaces to use asymmetric encryption and decryption primitives	[SWS_CRYPT_20750] [SWS_CRYPT_20751] [SWS_CRYPT_22700] [SWS_CRYPT_22701]
[RS_CRYPTO_02203]	The Crypto Stack shall provide interfaces to use message authentication code primitives	[SWS_CRYPT_20746] [SWS_CRYPT_22100] [SWS_CRYPT_22101] [SWS_CRYPT_22115] [SWS_CRYPT_22116] [SWS_CRYPT_50022] [SWS_CRYPT_50040]
[RS_CRYPTO_02204]	The Crypto Stack shall provide interfaces to use digital signature primitives	[SWS_CRYPT_20756] [SWS_CRYPT_20757] [SWS_CRYPT_23500] [SWS_CRYPT_23501] [SWS_CRYPT_23512] [SWS_CRYPT_24100] [SWS_CRYPT_24101] [SWS_CRYPT_24112] [SWS_CRYPT_50024]
[RS_CRYPTO_02205]	The Crypto Stack shall provide interfaces to use hashing primitives	[SWS_CRYPT_20747] [SWS_CRYPT_21100] [SWS_CRYPT_21101] [SWS_CRYPT_21115] [SWS_CRYPT_21116] [SWS_CRYPT_50059] [SWS_CRYPT_50060] [SWS_CRYPT_50115] [SWS_CRYPT_50116]
[RS_CRYPTO_02206]	The Crypto Stack shall provide interfaces to configure and use random number generation	[SWS_CRYPT_00502] [SWS_CRYPT_20741] [SWS_CRYPT_22900] [SWS_CRYPT_22901] [SWS_CRYPT_22911] [SWS_CRYPT_22912] [SWS_CRYPT_22914] [SWS_CRYPT_22915] [SWS_CRYPT_30098] [SWS_CRYPT_40990]
[RS_CRYPTO_02207]	The Crypto Stack shall provide interfaces to use authenticated symmetric encryption and decryption primitives	[SWS_CRYPT_20100] [SWS_CRYPT_20101] [SWS_CRYPT_20745]
[RS_CRYPTO_02208]	The Crypto Stack shall provide interfaces to use symmetric key wrapping primitives	[SWS_CRYPT_20743] [SWS_CRYPT_24000]
[RS_CRYPTO_02209]	The Crypto Stack shall provide interfaces to use asymmetric key encapsulation primitives	[SWS_CRYPT_20752] [SWS_CRYPT_20753] [SWS_CRYPT_21400] [SWS_CRYPT_21800] [SWS_CRYPT_21801]
[RS_CRYPTO_02301]	The Crypto Stack API shall provide a standardized header files structure	[SWS_CRYPT_20099] [SWS_CRYPT_40099]
[RS_CRYPTO_02302]	The Crypto Stack API shall support a streaming approach	[SWS_CRYPT_10701] [SWS_CRYPT_10710] [SWS_CRYPT_20313] [SWS_CRYPT_21113] [SWS_CRYPT_21115] [SWS_CRYPT_22110] [SWS_CRYPT_22111] [SWS_CRYPT_22113] [SWS_CRYPT_22115] [SWS_CRYPT_23618] [SWS_CRYPT_23625] [SWS_CRYPT_24715] [SWS_CRYPT_50013] [SWS_CRYPT_50019] [SWS_CRYPT_50020] [SWS_CRYPT_50021] [SWS_CRYPT_50022] [SWS_CRYPT_50023] [SWS_CRYPT_50027] [SWS_CRYPT_50035] [SWS_CRYPT_50037] [SWS_CRYPT_50038] [SWS_CRYPT_50039] [SWS_CRYPT_50040] [SWS_CRYPT_50041] [SWS_CRYPT_50044] [SWS_CRYPT_50045] [SWS_CRYPT_50046] [SWS_CRYPT_50047] [SWS_CRYPT_50048] [SWS_CRYPT_50049] [SWS_CRYPT_50057] [SWS_CRYPT_50058] [SWS_CRYPT_50067] [SWS_CRYPT_50068] [SWS_CRYPT_50069] [SWS_CRYPT_50070] [SWS_CRYPT_50073]

▽

△

Requirement	Description	Satisfied by
		△ [SWS_CRYPT_50074] [SWS_CRYPT_50075] [SWS_CRYPT_50076] [SWS_CRYPT_50077] [SWS_CRYPT_50078] [SWS_CRYPT_50079] [SWS_CRYPT_50080] [SWS_CRYPT_50081] [SWS_CRYPT_50087] [SWS_CRYPT_50088] [SWS_CRYPT_50089] [SWS_CRYPT_50090] [SWS_CRYPT_50139] [SWS_CRYPT_50140] [SWS_CRYPT_50152] [SWS_CRYPT_50153] [SWS_CRYPT_50158] [SWS_CRYPT_50159] [SWS_CRYPT_50176] [SWS_CRYPT_50177] [SWS_CRYPT_50192] [SWS_CRYPT_60016] [SWS_CRYPT_60017] [SWS_CRYPT_60018] [SWS_CRYPT_60019]
[RS_CRYPTO_02304]	The Crypto Stack API should support the possibility to move a state of a "counter mode" stream cipher to a random position	[SWS_CRYPT_23613]
[RS_CRYPTO_02305]	The Crypto Stack design shall separate cryptographic API from key access API	[SWS_CRYPT_00004] [SWS_CRYPT_10000] [SWS_CRYPT_10001] [SWS_CRYPT_20700] [SWS_CRYPT_41024] [SWS_CRYPT_50015] [SWS_CRYPT_50016] [SWS_CRYPT_60000] [SWS_CRYPT_60001] [SWS_CRYPT_60002] [SWS_CRYPT_60003]
[RS_CRYPTO_02306]	The Crypto Stack shall support integration with a Public Key Infrastructure (PKI)	[SWS_CRYPT_20618] [SWS_CRYPT_20620] [SWS_CRYPT_20907] [SWS_CRYPT_22501] [SWS_CRYPT_40001] [SWS_CRYPT_40099] [SWS_CRYPT_40101] [SWS_CRYPT_40111] [SWS_CRYPT_40112] [SWS_CRYPT_40115] [SWS_CRYPT_40151] [SWS_CRYPT_40152] [SWS_CRYPT_40153] [SWS_CRYPT_40154] [SWS_CRYPT_40155] [SWS_CRYPT_40156] [SWS_CRYPT_40157] [SWS_CRYPT_40158] [SWS_CRYPT_40159] [SWS_CRYPT_40202] [SWS_CRYPT_40203] [SWS_CRYPT_40213] [SWS_CRYPT_40214] [SWS_CRYPT_40215] [SWS_CRYPT_40216] [SWS_CRYPT_40300] [SWS_CRYPT_40301] [SWS_CRYPT_40400] [SWS_CRYPT_40401] [SWS_CRYPT_40411] [SWS_CRYPT_40600] [SWS_CRYPT_40601] [SWS_CRYPT_40604] [SWS_CRYPT_40614] [SWS_CRYPT_40616] [SWS_CRYPT_40621] [SWS_CRYPT_40622] [SWS_CRYPT_40626] [SWS_CRYPT_40627] [SWS_CRYPT_40640] [SWS_CRYPT_40641] [SWS_CRYPT_40802] [SWS_CRYPT_41019] [SWS_CRYPT_50003] [SWS_CRYPT_50093] [SWS_CRYPT_50094] [SWS_CRYPT_50095] [SWS_CRYPT_50096] [SWS_CRYPT_50097] [SWS_CRYPT_50098] [SWS_CRYPT_50099] [SWS_CRYPT_50100] [SWS_CRYPT_50101] [SWS_CRYPT_50102] [SWS_CRYPT_50103] [SWS_CRYPT_50104] [SWS_CRYPT_50105] [SWS_CRYPT_50106] [SWS_CRYPT_50107] [SWS_CRYPT_50108] [SWS_CRYPT_50109] [SWS_CRYPT_50110] [SWS_CRYPT_50111] [SWS_CRYPT_50112] [SWS_CRYPT_50113] [SWS_CRYPT_50114] [SWS_CRYPT_50117] [SWS_CRYPT_50118] [SWS_CRYPT_50119] [SWS_CRYPT_50120] [SWS_CRYPT_50121] [SWS_CRYPT_50122] [SWS_CRYPT_50123] [SWS_CRYPT_50124] [SWS_CRYPT_50125] [SWS_CRYPT_50126] ▽

▽

△

Requirement	Description	Satisfied by
		△ [SWS_CRYPT_50127] [SWS_CRYPT_50128] [SWS_CRYPT_50129] [SWS_CRYPT_50130] [SWS_CRYPT_50131] [SWS_CRYPT_50179] [SWS_CRYPT_50180] [SWS_CRYPT_50181] [SWS_CRYPT_50182] [SWS_CRYPT_50183] [SWS_CRYPT_50184] [SWS_CRYPT_50185] [SWS_CRYPT_50186] [SWS_CRYPT_50187] [SWS_CRYPT_50191] [SWS_CRYPT_50213] [SWS_CRYPT_50214]
[RS_CRYPTO_02307]	The Crypto Stack design shall separate cryptographic API from the PKI API	[SWS_CRYPT_20700]
[RS_CRYPTO_02308]	The Crypto Stack shall support a unified cryptographic primitives naming convention, common for all suppliers	[SWS_CRYPT_03910] [SWS_CRYPT_60004]
[RS_CRYPTO_02309]	The Crypto Stack API shall support the run-time configurable usage style	[SWS_CRYPT_20103] [SWS_CRYPT_29004] [SWS_CRYPT_50025] [SWS_CRYPT_50028] [SWS_CRYPT_50029] [SWS_CRYPT_50030] [SWS_CRYPT_50031] [SWS_CRYPT_50032] [SWS_CRYPT_50033] [SWS_CRYPT_50042] [SWS_CRYPT_50050] [SWS_CRYPT_50061] [SWS_CRYPT_50062] [SWS_CRYPT_50063] [SWS_CRYPT_50064] [SWS_CRYPT_50065] [SWS_CRYPT_50071] [SWS_CRYPT_50082] [SWS_CRYPT_50091] [SWS_CRYPT_50133] [SWS_CRYPT_50134] [SWS_CRYPT_50135] [SWS_CRYPT_50141] [SWS_CRYPT_50142] [SWS_CRYPT_50143] [SWS_CRYPT_50144] [SWS_CRYPT_50154] [SWS_CRYPT_50160] [SWS_CRYPT_50161] [SWS_CRYPT_50162] [SWS_CRYPT_50163] [SWS_CRYPT_50164] [SWS_CRYPT_50165] [SWS_CRYPT_50178]
[RS_CRYPTO_02310]	The Crypto Stack API shall support an efficient mechanism of error states notification	[SWS_CRYPT_10099] [SWS_CRYPT_19902] [SWS_CRYPT_19903] [SWS_CRYPT_19904] [SWS_CRYPT_19905] [SWS_CRYPT_19906] [SWS_CRYPT_19950] [SWS_CRYPT_19951] [SWS_CRYPT_19952] [SWS_CRYPT_19953] [SWS_CRYPT_19954] [SWS_CRYPT_41050]
[RS_CRYPTO_02401]	The Crypto Stack should support a joint usage of multiple back-end cryptography providers including ones with non-extractable keys	[SWS_CRYPT_20098] [SWS_CRYPT_20099] [SWS_CRYPT_20700] [SWS_CRYPT_30001] [SWS_CRYPT_30002] [SWS_CRYPT_30003] [SWS_CRYPT_30403] [SWS_CRYPT_30404] [SWS_CRYPT_30406] [SWS_CRYPT_41055]
[RS_CRYPTO_02403]	The Crypto Stack shall support isolating keys and requests	[SWS_CRYPT_22500] [SWS_CRYPT_23800]
[RS_CRYPTO_02405]	The Crypto Stack shall support the key slots identification in a way independent from a concrete deployment	[SWS_CRYPT_41058] [SWS_CRYPT_50145] [SWS_CRYPT_50146] [SWS_CRYPT_50151] [SWS_CRYPT_50166] [SWS_CRYPT_50167] [SWS_CRYPT_50172] [SWS_CRYPT_60010] [SWS_CRYPT_60011] [SWS_CRYPT_60020]
[RS_Ids_00810]	Basic SW security events	[SWS_CRYPT_41022] [SWS_CRYPT_41059] [SWS_CRYPT_41060] [SWS_CRYPT_41061] [SWS_CRYPT_41062] [SWS_CRYPT_41063] [SWS_CRYPT_41064] [SWS_CRYPT_41065] [SWS_CRYPT_41066]

Table 6.1: Requirements Tracing

7 Functional specification

The AUTOSAR Adaptive architecture organizes the software of the AUTOSAR Adaptive foundation as Functional Clusters. These clusters offer common functionality as services to the applications. The Functional Cluster Cryptography (FC Crypto) is part of the AUTOSAR Adaptive Platform Foundation.

The FC Crypto provides the infrastructure to access multiple implementations of cryptographic operations through a standardized interface. It provides access to cryptographic primitives via the Crypto Provider and access to certificate management via the X.509 Certificate Management Provider. It supports common automotive security use-cases like secure communication (e.g., SecOC, TLS), authenticated diagnostics (e.g, service 0x29), and secure software update.

This specification includes the syntax and functional behavior of the API and its relationship to the model.

7.1 Architecture & Concepts

In this section the architecture of Crypto and several concepts used by Crypto are described.

7.1.1 Architecture

The FC Crypto offers applications and other Adaptive AUTOSAR Functional Clusters a standardized interface to perform cryptographic operations, manage cryptographic keys, and handle certificate. The interfaces defined by FC Crypto are designed to enable integraton of 3rd party cryptographic libraries and hardware-based elements (e.g., HSMs). This facilitates implementation of a security ”trust anchor” or acceleration of cryptographic transformations.

Figure 7.1 depicts the high-level architecture of FC Crypto.

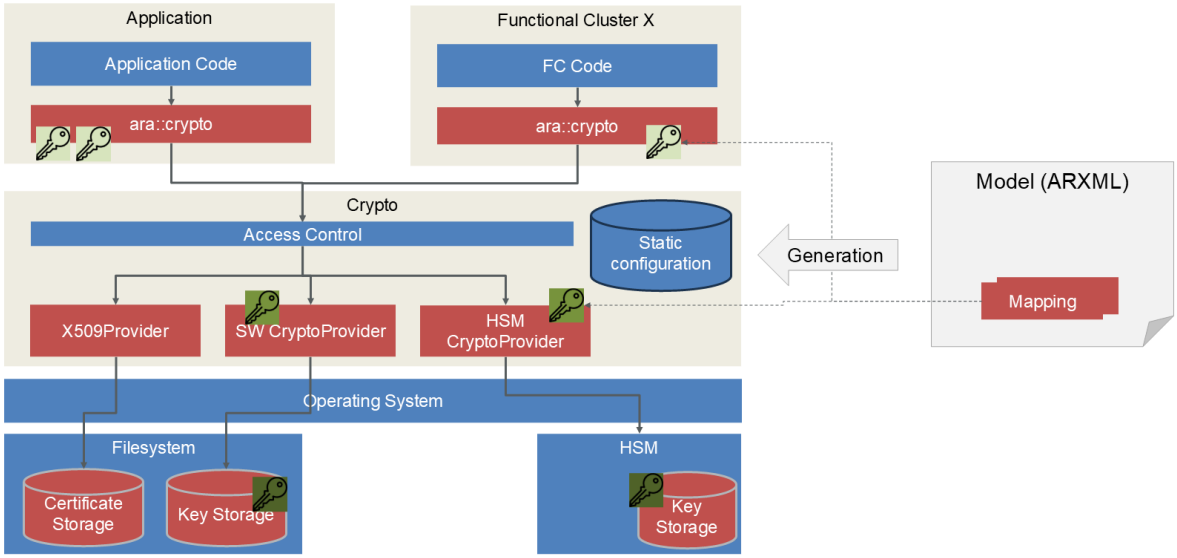

Figure 7.1: Exemplary High-level `FC Crypto` architecture

A CryptoProvider provides access to the implementation of a set of cryptographic primitives and a key storage. Typically, an cryptographic module within a trusted execution environment is providing the actual implementation of primitives and key storage, and the CryptoProvider is providing a wrapper that allows access to different implementations using the same interface. A CryptoProvider is implemented by a stack vendor or a third party by implementing the `ara::crypto::cryp::CryptoProvider` class, as well as the classes for cryptographic contexts and the classes for cryptographic keys. It is possible to have multiple different CryptoProviders instantiated on one machine. However, although the interfaces are the same across multiple CryptoProvider implementations, there is no interoperability between different CryptoProviders. For instance, it is not possible to use a symmetric key that belongs to one CryptoProvider for encryption in another CryptoProvider. This is not a limitation of ara::crypto but an inherent and desired property of cryptographic modules.

Every key managed by ara::crypto is managed by exactly one CryptoProvider. A `ara::crypto::cryp::KeySlot` represents the interface from the application to access and use a persisted key. Each KeySlot is statically mapped to one CryptoProvider during deployment. Access to KeySlots is provided to applications through modelled ports using InstanceSpecifiers.

The X.509 Provider provides access to persisted certificates and supports certificate management and verification functionality. In contrast to the CryptoProvider, there exists exactly one X.509 Provider instance on one machine.

7.1.2 Usage of Crypto

7.1.2.1 Usage of Crypto by Applications

This section describes the required modelling during the design and deployment phase of an adaptive application. During design, the application designer has to decide, which persisted cryptographic keys, which persisted certificates, and which CryptoProviders the application will use.

- For each persisted key that will be used and modified by an application (e.g., the key is generated by the application and later used for signature generation), the application designer creates a `CryptoKeySlotInterface`.
- For each persisted key that will be used by an application, but modified only by a different application, the application designer creates a `CryptoKeySlotClientInterface`.
- For each persisted certificate that will be used by an application, the application designer creates a `CryptoCertificateInterface`.
- For each group of persisted certificates that will be used by an application, the application designer creates a `CryptoCertificateGroupInterface`.
- For each CryptoProvider that will be used by an application independently of cryptographic keys (e.g., to calculate cryptographic hash values), the application designer creates a `CryptoProviderInterface`.

Additionally, the application designer has to model the key types (e.g., AES-128, ed25519 private key) the application expects in a key slot. The application developer can use each of the modelled interfaces by passing the InstanceSpecifier representing the modelled PortInterface to an apropriate function of ara::crypto.

During integration, each interface has to be mapped to a resource, e.g., each `CryptoKeySlotInterface` needs to be mapped to a `CryptoKeySlot`, which represents the actual key slot on the machine. Multiple `CryptoKeySlotInterface`'s may be mapped to the same `CryptoKeySlot`, meaning that multiple applications can access the same key slot. The integrator has to ensure that the desired properties modeled in the PortInterfaces match the actual properties of the resource on the machine, e.g., that a KeySlot that an application expects to only contain an AES-128 key does not contain an ECDSA key.

7.1.2.2 Usage of Crypto by other Functional Clusters

The implementation of some `Adaptive Platform Services` such as update and configuration, communication, persistency, or diagnostics also require access to cryptographic services and cryptographic keys to fulfill their functionality. Such access is modelled using cross functional cluster mappings. For instance, if a key is being used by SecOC for protection of a PDU, the model element `ComSecOcToCryptoKeySlotMapping` is used to configure the KeySlot to be used by SecOC. A de-

tailled description of the available model can be found in the TPS Manifest in Chapter Cross-functional cluster interaction.

Typically, such key material is only used by a functional cluster, but managed (e.g., imported, generated, updated) by an adaptive application. In this case, the mechanisms described in Section 7.1.2.1 are used.

7.1.3 Access Control

Crypto supports implementing the least privilege principle by restricting, which application can use and modify cryptographic keys and certificates. The policy, i.e., which application can access which resource in which way, is modelled in the manifest and enforced at runtime. In order to enforce these access restrictions, a separation between the application process using ara::crypto and the policy enforcement is required. This separation can be implemented using, e.g., OS mechanisms such as process isolation.

Access to a KeySlot is limited to certain processes in the manifest using the `CryptoKeySlotToPortPrototypeMapping` and `CryptoKeySlotToClientPortPrototypeMapping` model elements:

[SWS_CRYPT_10000] Access Control on Loading KeySlots

Upstream requirements: RS_CRYPTO_02004, RS_CRYPTO_02305

⌈`Crypto` shall restrict the processes that can load a `CryptoKeySlot` to those `Processes` referenced by a `CryptoKeySlotToPortPrototypeMapping` or a `CryptoKeySlotToClientPortPrototypeMapping` in the role `process`, which also reference the `CryptoKeySlot` in the role `keySlot`. Any unauthorized access attempt shall be treated as a violation.⌋

Write access to a KeySlot is limited to certain processes in the manifest using the `CryptoKeySlotToPortPrototypeMapping` model element:

[SWS_CRYPT_10001] Access Control on Modifying KeySlots

Upstream requirements: RS_CRYPTO_02004, RS_CRYPTO_02305

⌈`Crypto` shall restrict the processes that can modify the content of a `CryptoKeySlot` to those `Processes` referenced by a `CryptoKeySlotToPortPrototypeMapping` in the role `process`, which also references the `CryptoKeySlot` in the role `keySlot` and a `RPortPrototype` of type `CryptoKeySlotInterface` in the role `portPrototype`. Any unauthorized access attempt shall be treated as a violation.⌋

Access to a persisted Certificate is limited to certain processes in the manifest using the `CryptoCertificateToPortPrototypeMapping` model element:

[SWS_CRYPT_60000] Access Control on Loading Certificates

Upstream requirements: RS_CRYPTO_02004, RS_CRYPTO_02305

⌈Crypto shall restrict the processes that can access the content of a CryptoCertificate to those Processes referenced by a CryptoCertificateToPortPrototypeMapping in the role process, which also references the CryptoCertificate in the role cryptoCertificate and a RPortPrototype of type CryptoCertificateInterface in the role portPrototype. Any unauthorized access attempt shall be treated as a violation.⌋

Write access to a persisted Certificate is limited to certain processes in the manifest using the CryptoCertificateToPortPrototypeMapping model element and its attribute writeAccess:

[SWS_CRYPT_60001] Access Control on Modifying Certificates

Upstream requirements: RS_CRYPTO_02004, RS_CRYPTO_02305

⌈Crypto shall restrict the processes that can modify the content of a CryptoCertificate to those Processes referenced by a CryptoCertificateToPortPrototypeMapping in the role process, which also references the CryptoCertificate in the role cryptoCertificate and a RPortPrototype of type CryptoCertificateInterface in the role portPrototype, and has the attribute writeAccess set to True. Any unauthorized access attempt shall be treated as a violation.⌋

Access to a persisted CertificateGroup is limited to certain processes in the manifest using the CryptoCertificateGroupToPortPrototypeMapping model element:

[SWS_CRYPT_60002] Access Control on Loading Certificate Groups

Upstream requirements: RS_CRYPTO_02004, RS_CRYPTO_02305

⌈Crypto shall restrict the processes that can access the content of a CryptoCertificateGroup to those Processes referenced by a CryptoCertificateGroupToPortPrototypeMapping in the role process, which also references the CryptoCertificateGroup in the role certificateGroup and a RPortPrototype of type CryptoCertificateGroupInterface in the role portPrototype. Any unauthorized access attempt shall be treated as a violation.⌋

[SWS_CRYPT_60003] Access Control on Modifying Certificate Groups

Upstream requirements: RS_CRYPTO_02004, RS_CRYPTO_02305

⌈Crypto shall restrict the processes that can modify the content of a CryptoCertificateGroup to those Processes referenced by a CryptoCertificateGroupToPortPrototypeMapping in the role process, which also references the CryptoCertificateGroup in the role certificateGroup and a RPortPrototype of type CryptoCertificateGroupInterface in the role portPrototype, and has the attribute writeAccess set to True. Any unauthorized access attempt shall be treated as a violation.⌋

[SWS_CRYPT_41056] Log Access not granted

Upstream requirements: RS_AP_00158, RS_CRYPTO_02110

⌈Whenever an application attempts to access a crypto resource but it is unauthorized, `FC Crypto` shall log a DltMessage of type ResourceAccessNotGranted with arguments set to

- ProcessId: Process identifier of the process that was not granted access to a crypto resource.
- ResourceInstanceSpecifier: A Provider or KeySlot or any crypto resource instance specifier that the process tried to access without permission.

⌋

7.1.4 Usage Control

When a key is generated, imported, or derived, the application can limit how the key can be used afterwards. The type `ara::crypto::AllowedUsageFlags` is used to represent the allowed usages of a key as a bitmask. Table 7.1 gives an overview of the available allowed usages. Multiple flags can be combined with a bitwise *or* in order to allow several usages of a key.

Allowed Usage Flag	***Applicable Key Type***	***Description***	***Relevant Contexts/Functions***
`kAllowDataEncryption`	SymmetricKey, PublicKey	The key can be used for data encryption	SymmetricCipherCtx, AuthCipherCtx, EncryptorPublicCtx
`kAllowDataDecryption`	SymmetricKey, PrivateKey	The key can be used for data decryption.	SymmetricCipherCtx, AuthCipherCtx, DecryptorPrivateCtx
`kAllowMac`	SymmetricKey	The key can be used for MAC generation and verification.	MessageAuthnCodeCtx
`kAllowSigGeneration`	PrivateKey	The key can be used for signature generation.	SignerPrivateCtx
`kAllowSigVerification`	PublicKey	The key can be used for signature verification.	VerifierPublicCtx
`kAllowKeyAgreement`	PrivateKey, PublicKey	The key can be used as private or public input to key agreement.	KeyAgreementPrivateCtx

▽

△

kAllowKeyDerivation	SymmetricKey	The key can be used for key derivation	KeyDerivationCtx
kAllowHashCalculation	SymmetricKey, PrivateKey, PublicKey	The key can be used as an input for hashing	HashFunctionCtx
kAllowKeyWrapping	SymmetricKey, PublicKey	The key can be used as a key encryption key for symmetric key wrapping or asymmetric key encapsulation.	SymmetricKeyWrapperCtx, KeyEncapsulatorPublicCtx
kAllowKeyUnwrapping	SymmetricKey, PrivateKey	The key can be used as a key encryption key for symmetric key unwrapping or asymmetric key decapsulation.	SymmetricKeyWrapperCtx, KeyDecapsulatorPrivateCtx
kAllowPersist	SymmetricKey, PrivateKey, PublicKey	The key can persisted.	CryptoObject::Save
kAllowPlaintextExport	SymmetricKey, PrivateKey, PublicKey	The key can be exported.	Serializable::ExportPublicly
kAllowSecuredExport	SymmetricKey, PrivateKey, PublicKey	The key can be exported.	SymmetricKeyWrapperCtx, KeyEncapsulatorPrivateCtx
kAllowRngInit	SymmetricKey	The symmetric key can be used for seeding of a RandomGeneratorCtx	RandomGeneratorCtx
kAllowDerivedPlaintextExport	SymmetricKey	The symmetric key can be used for key derivation of a key that is exportable in plaintext	KeyDerivationCtx

Table 7.1: Allowed Usage Flags

7.1.5 Identification of Cryptographic Transformations and Key Types

The type of all keys and cryptographic transformations must be unambiguously identifiable to enable the application designer and developer to unambiguously express the desired key types and transformations to by used by the CryptoProvider.

The type of a key is typically defined by an algorithm (e.g., AES, RSA, ed25519) and optionally by a length (e.g., 2048 bit) or curve (e.g., secp256r1), and whether the key is a private or public key. A cryptographic transformation is typically defined by several elements:

- The operation that is performed by the transformation, e.g., hashing, encryption, signature verification.
- The algorithm that forms the basis of the transformation, e.g., SHA2-256, AES-128, ed25519.
- An optional variant that specifies how the algorithm is applied. This includes the mode (e.g., CBC), the signature scheme (e.g., RSAES-OAEP), the used curve (e.g., secp256r1), and the number of rounds.
- An optional parameter that specifies, e.g., an additional algorithm that is executed as part of the transformation (e.g., SHA2-256, Poly1305) or padding (e.g., PKCS7).

In this specification, a string based identification of transformations and key types is used to enable future extensions with additional transformations or algorithms. In the API, the type `ara::crypto::AlgId` is used to identify a tranformation. The AlgId can be passed by an application when creating a transformation context. Additionally, it is used when generating or importing a key. It is also used as part of the manifest to specify the key type of a KeySlot.

The AlgId string has to adhere to the to following syntax:

[SWS_CRYPT_03910] Syntax of AlgId

Upstream requirements: RS_CRYPTO_02308

⌈The AlgId string shall adhere to the following syntax specified in ABNF (RFC 5234):

AlgId = 1*allowedCharacters ["/" AlgId]

allowedCharacters = ALPHA / DIGIT / "-" / "." / "_" / "," / "+"⌋

[SWS_CRYPT_60004] Handling of Invalid AlgId

Upstream requirements: RS_CRYPTO_02308

⌈Every function of ara::crypto that takes a AlgId as parameter shall return kUnknownIdentifier if the AlgId string does not satisfy the syntax specified in [SWS_CRYPT_03910] or if the AlgId is not supported by the function.⌋

A key is typically characterized by fewer parameters than a cryptographic transformation. Therefore, the AlgId used for identifying a key can be a substring of the AlgId used for identifying a transformation (e.g., "AES-128" is sufficient for generating a key, whereas "AES-128/CBC/PKCS7" specifies a transformation). However, there can be cases where even for key generation more information is required, e.g., because the underlying HSM enforces a single-use principle and requires knowledge of the transformation in which the key will be used later.

The vendor of a CryptoProvider documents the available AlgId's, which can be then used by an application. Appendix D lists proposed names for a set of standardized transformations. A CryptoProvider that implements a transformation that is contained in Appendix D should use the proposed names. However, a CryptoProvider implementation may implement only a subset of the listed transformations or it may use additional AlgIds for transformations that are not contained in Appendix D.

7.1.6 Concurrent Access

While unlimited concurrent access to ara::crypto is theoreticaly possible, in reality there are only limited resources available. For instance, an HSM might only have a certain number of AES accelerators, limiting the number of operations that can be processed in parallel. In order to represent this behavior, all functions of ara::crypto, which may need to allocate resources, may return kBusyResource in case the requested operation temporarly cannot be performed because a required resource is busy. This includes all functions for key generation, loading keys, importing keys, the Start() function of transformation contexts' streaming interface, and the single-call functions of transformation contexts.

[SWS_CRYPT_60005] Limit concurrent access to internal resources

Upstream requirements: RS_CRYPTO_02004

⌈All functions of ara::crypto shall return kBusyResource if the requested operation temporarly cannot be performed because a required resource is busy.⌋

The implementation of a CryptoProvider may perform internal scheduling or sequentializing of operations. For instance, consider a HSM which can only perform one MAC calculation at a time, but two small single-call MAC calculations are triggered at nearly the same time. Because these operations are fast to perform, the CryptoProvider may chose to perform both operations after each other, without the need to return kBusyResource. However, if a streaming operation is ongoing and another operation can only be performed once the ongoing streaming operation is finished, returning kBusyResource might be the more appropriate choice instead of blocking the second call for an indefinite amount of time.

Another issue with concurrent access is that the modification of a KeySlot during an ongoing operation would lead to a faulty operation. While the modification and use of the key can be performed sequentially by the CryptoProvider in the case of single-call operations, this is not possible for streaming operations. Therefore, Crypto prevents the modification of KeySlot content which is currently in use by a streaming operation.

[SWS_CRYPT_60006] Prevent modification of currently used KeySlots

Upstream requirements: RS_CRYPTO_02004

⌈The functions `ara::crypto::cryp::WriteableKeySlot::Clear` and `ara::crypto::cryp::CryptoObject::Save` shall return kBusyResource if the key that

is currently stored in the KeySlot is being used in an ongoing streaming operation, i.e., between the Start() and Finish() function of a context.⌋

7.2 Functional Cluster Lifecycle

7.2.1 Startup

Using `ara::core::Initialize` and `ara::core::Deinitialize`, the application can initialize and de-initialize `FC Crypto` resources allocated to the application.

[SWS_CRYPT_00101]

Upstream requirements: RS_CRYPTO_02110

⌈When `ara::core::Initialize` is called, the `FC Crypto` shall read in the manifest information and prepare the access structures that are defined in the manifest.⌋

Hint: Access structures may encompass the communication channel between the application process and the stack process or other resource required by the `CryptoAPI`.

[SWS_CRYPT_41051] Log initialization failed

Upstream requirements: RS_CRYPTO_02110

⌈Whenever Initialization of crypto structures failed during a call of `ara::core::Initialize`, `FC Crypto` shall log a DltMessage of type InitializationFailed with arguments set to:

- ProcessId: The process failed to initialize crypto

⌋

7.2.2 Shutdown

[SWS_CRYPT_00102]

Upstream requirements: RS_CRYPTO_02004, RS_CRYPTO_02007, RS_CRYPTO_02102

⌈When `ara::core::Deinitialize` is called, the `FC Crypto` shall ensure that all open contexts are closed and all occupied ressources are freed.⌋

[SWS_CRYPT_00103]

Upstream requirements: RS_CRYPTO_02004

⌈When `ara::core::Deinitialize` is called, the `FC Crypto` shall ensure that all associated persist operations in this context of this application are executed successfully and no new persist operations are started.⌋

[SWS_CRYPT_41052] Log deInitialization failed

Upstream requirements: RS_CRYPTO_02102

⌈Whenever deInitialization of crypto structures failed during call of `ara::core::Deinitialize`, `FC Crypto` shall log a DltMessage of type DeInitializationFailed with arguments set to:

- ProcessId: The process failed to deInitialize crypto

⌋

Note: the application is expected not to call any API of `FC Crypto` before `ara::core::Initialize` or after `ara::core::Deinitialize`. According to SWS_CORE_90021, such behavior will be trated as a violation.

[SWS_CRYPT_41050] Initialization violation

Upstream requirements: RS_CRYPTO_02310

⌈The function `ara::crypto::LoadX509Provider` shall check that `ara::core::Initialize` has been called successfully and `ara::core::Deinitialize` has not yet been executed. If the check failed, this shall be treated as a violation according to [SWS_CORE_00021].⌋

7.3 Crypto Provider

A `Crypto Provider` is a structural element that organizes `Cryptographic primitives`. Every `Crypto Provider` typically represents either one hardware element, e.g., trusted platform module (TPM) or hardware security module (HSM), or one software element, e.g., cryptographic library.

[SWS_CRYPT_00004] Functionality of a CryptoProvider

Upstream requirements: RS_CRYPTO_02305

⌈Each derived implementation of the interface class `ara::crypto::cryp::CryptoProvider` shall encapsulate cryptographic transformations and associated resources, such as `ara::crypto::cryp::CryptoObject` and `Cryptographic primitives`, of a single software or hardware cryptography implementation.⌋

Note: a `Crypto Provider` may expose only a subset of all available transformations or primitives of the underlying software or hardware cryptography implementation (e.g. in case of weak or outdated primitives). However, this implementation detail shall be documented and communicated to the user.

[SWS_CRYPT_00007]

Upstream requirements: RS_CRYPTO_02001, RS_CRYPTO_02002

⌈Derived implementations of the interface class `ara::crypto::cryp::CryptoProvider` shall isolate all non-session `Key Material` from the user (`Adaptive Application`).⌋

7.3.1 KeySlot Access

An application can use the InstanceSpecifier representing a modeled `CryptoKeySlotInterface` or `CryptoKeySlotClientInterface` to access a KeySlot. If a `CryptoKeySlotClientInterface` is used, the key stored in the KeySlot can only be used but not modified. Usage of a `CryptoKeySlotInterface` additionally allows modification of the key stored in the KeySlot. The class `CryptoKeySlotToPortPrototypeMapping` defines the mapping between a `Process`, a `CryptoKeySlot`, and an `RPortPrototype`.

If a KeySlot is only accessed in a read-only manner, the function `ara::crypto::LoadKeySlot` can be used to get a KeySlot object that is read-only.

[SWS_CRYPT_60020] LoadKeySlot

Upstream requirements: RS_CRYPTO_02405

⌈The function `ara::crypto::LoadKeySlot` shall return the KeySlot instance that represents the key slot resource identified by the provided `ara::core::InstanceSpecifier` representing a modeled `CryptoKeySlotInterface` or `CryptoKeySlotClientInterface` through the `CryptoKeySlotToPortPrototypeMapping`.⌋

If a KeySlot will also be modified, the function `ara::crypto::LoadWriteableKeySlot` can be used to get a WriteableKeySlot object, which can be used to modify the content of the KeySlot.

[SWS_CRYPT_60010] LoadWriteableKeySlot

Upstream requirements: RS_CRYPTO_02405

⌈The function `ara::crypto::LoadWriteableKeySlot` shall return the WriteableKeySlot instance that represents the key slot resource identified by the provided `ara::core::InstanceSpecifier` representing a modeled `CryptoKeySlotInterface` through the `CryptoKeySlotToPortPrototypeMapping`.⌋

A KeySlot provides several functions, which can be used by an application to query information about the KeySlot.

Each KeySlot is statically configured to contain either a symmetric, or a public, or a private key. Additionally, the type of the stored key is statically configured using the AlgId described in Section 7.1.5. The AlgId of a KeySlot is typically less specific than the AlgId used for the actual transformation. For example, the AlgId of a KeySlot might

be "AES-128" and the stored key is later used for encryption with a AlgId of "AES-128/CBC/PKCS7".

In order to delete a previously persisted key, the content of a WriteableKeySlot can be cleared using the function `ara::crypto::cryp::WriteableKeySlot::Clear`.

[SWS_CRYPT_60011] Deletion of a persisted key

Upstream requirements: RS_CRYPTO_02405

⌈The function `ara::crypto::cryp::WriteableKeySlot::Clear` shall securely delete the content of the KeySlot, i.e., the stored key. If the key is currently being used in an ongoing operation in streaming mode, the function shall return kBusyResource.⌋

[SWS_CRYPT_41058] Log KeySlot content changed

Upstream requirements: RS_CRYPTO_02405

⌈Whenever a keyslot content is changed, i.e., added, modified or deleted, `FC Crypto` shall log a DltMessage of type KeySlotContentChanged with arguments set to:

- KeySlotInstanceSpecifier: The key slot instance specifier.

⌋

7.3.2 CryptoProvider Instantiation

Crypto provides two functions to instantiate a CryptoProvider. First, if a `CryptoProviderInterface` is modelled, the function `ara::crypto::LoadCryptoProvider` can be used to instantiate a CryptoProvider. The main use cases for such a direct instantiation are, e.g., the use of keyless operations such as hashing or the generation and use of session keys. Secondly, if the application accesses modelled KeySlots as described in Section 7.3.1, the function `ara::crypto::cryp::KeySlot::GetCryptoProvider` can be used to instantiate the correct CryptoProvider for subsequent use of the stored key.

[SWS_CRYPT_41055] Log Crypto Provider loading failed

Upstream requirements: RS_CRYPTO_02401

⌈Whenever Loading a `Crypto Provider` failed when calling `ara::crypto::LoadCryptoProvider`, `FC Crypto` shall log a DltMessage of type CryptoProviderLoadingFailed with arguments set to:

- CryptoProviderInstanceSpecifier: `Crypto Provider` instance specifier.

⌋

7.3.3 Runtime Key Instantiation and Usage

After loading a KeySlot, the KeySlot object can either be loaded into a CryptoProvider in order to obtain a runtime CryptoObject, or it can be used directly in a transformation context in order to specify the key used for the cryptographic transformation (cf. 7.4).

A runtime key object can be creating using the functions `ara::crypto::cryp::CryptoProvider::LoadSymmetricKey`, `ara::crypto::cryp::CryptoProvider::LoadPublicKey`, and `ara::crypto::cryp::CryptoProvider::LoadPrivateKey`.

A runtime key can be stored in a KeySlot that is managed by the same CryptoProvider and has the same AlgId and CryptoObjectType as the runtime key using the function `ara::crypto::cryp::CryptoObject::Save`.

[SWS_CRYPT_60012] Save runtime key to a KeySlot

Upstream requirements: RS_CRYPTO_02004

⌈The function `ara::crypto::cryp::CryptoObject::Save` shall verify that the AllowedUsageFlags include kAllowPersist, verify that the KeySlot is compatible to the key, verify that the KeySlot is writeable by the application, verify that the key currently stored in the KeySlot is not being used in an ongoing streaming operation, and store the key represented by the CryptoObject in the provided KeySlot.⌋

Note: Write access to a KeySlot is defined in [SWS_CRYPT_10001].

7.3.3.1 Utility functions

The classes `ara::crypto::cryp::SymmetricKey`, `ara::crypto::cryp::PublicKey`, and `ara::crypto::cryp::PrivateKey` provide the function `ara::crypto::cryp::CryptoObject::GetAlgId` to query the AlgId of the key. The classes `ara::crypto::cryp::SymmetricKey`, `ara::crypto::cryp::PublicKey`, and `ara::crypto::cryp::PrivateKey` provide the function `ara::crypto::cryp::CryptoObject::GetAllowedUsage` to query the AllowedUsage (cf. 7.1.4) of the key. A `ara::crypto::cryp::PrivateKey` provides the function `ara::crypto::cryp::PrivateKey::GetPublicKey` that allows to query the corresponding public key.

7.3.4 Key Generation

In order to generate new keys, a CryptoProvider provides the functions `ara::crypto::cryp::CryptoProvider::GenerateSymmetricKey` and `ara::crypto::cryp::CryptoProvider::GeneratePrivateKey`. An application can pass a AlgId identifying the type of the key and the allowed usages of the generated key to the functions.

[SWS_CRYPT_40962]

Upstream requirements: RS_CRYPTO_02101

⌈The functions `ara::crypto::cryp::CryptoProvider::GeneratePrivateKey` and `ara::crypto::cryp::CryptoProvider::GenerateSymmetricKey` shall generate secret key-material according to the provided `ara::crypto::AlgId` and return an instance of `ara::crypto::cryp::PrivateKey` or `ara::crypto::cryp::SymmetricKey`, respectively. Each function shall initialize the object according to the provided `ara::crypto::AllowedUsageFlags`.⌋

7.3.5 Plaintext Key Management

7.3.5.1 Plaintext Key Import

Key material can be imported into a CryptoProvider in plaintext form and in protected form. The functions `ara::crypto::cryp::CryptoProvider::ImportPublicKey`, `ara::crypto::cryp::CryptoProvider::ImportSymmetricKey`, and `ara::crypto::cryp::CryptoProvider::ImportPrivateKey` can be used to import plaintext key material. When importing a key, the application can set the allowed usage of the imported key.

Depending on the key material and the context, importing or exporting confidential plaintext key material may represent a security risk. Therefore, the use of plaintext key import or export should be evaluated carefully by all involved parties. Additionally, a CryptoProvider may not support plaintext import or export of certain key types (e.g., symmetric or private keys) for security reasons.

[SWS_CRYPT_04205] Import of public keys

Upstream requirements: RS_CRYPTO_02004

⌈The function `ara::crypto::cryp::CryptoProvider::ImportPublicKey` shall deserialize the provided serialized data from the format specified using the formatId parameter. The function shall create a PublicKey object with the AlgId passed as a parameter or, if the serialized data contains that information and no AlgId is passed as a parameter, based on the serialized data. The function shall return the created key object.⌋

[SWS_CRYPT_60013] Import of private keys

Upstream requirements: RS_CRYPTO_02004

⌈The function `ara::crypto::cryp::CryptoProvider::ImportPrivateKey` shall deserialize the provided serialized data from the format specified using the formatId parameter. The function shall create a PrivateKey object with the AlgId passed as a parameter or, if the serialized data contains that information and no AlgId is passed as a parameter, based on the serialized data. The function shall return the created key object.⌋

[SWS_CRYPT_60014] Import of symmetric keys

Upstream requirements: RS_CRYPTO_02004

⌈The function `ara::crypto::cryp::CryptoProvider::ImportSymmetricKey` shall deserialize the provided serialized data from the format specified using the formatId parameter. The function shall create a SymmetricKey object with the AlgId passed as a parameter or, if the serialized data contains that information and no AlgId is passed as a parameter, based on the serialized data. The function shall return the created key object.⌋

[SWS_CRYPT_60015] Storage of imported Keys

Upstream requirements: RS_CRYPTO_02004

⌈The function `ara::crypto::cryp::CryptoObject::Save` shall return `kIncompatibleObject` if the PrivateKey or SymmetricKey to be saved has been imported in plaintext format and for the `CryptoKeySlotToPortPrototypeMapping` referencing the `Process` which called the function `ara::crypto::cryp::CryptoObject::Save` and referencing the `CryptoKeySlot` in which the key shall be stored the parameter `allowPlainTextImport` is not configured as True.⌋

7.3.5.2 Plaintext Key Export

The function `ara::crypto::Serializable::ExportPublicly` provides a way to export symmetric, private, and public keys.

[SWS_CRYPT_04203] Exporting public objects

Upstream requirements: RS_CRYPTO_02004

⌈The function `ara::crypto::Serializable::ExportPublicly` shall serialize the object in the format specified using the formatId parameter, write the serialized data into the provided output data buffer, and return the size of the written serialized data. If the function is a member function of a PublicKey, PrivateKey, or SymmetricKey, it shall additionally verify prior to serialization that its allowed usage includes `kAllowPlainTextExport` and the parameter `allowPlainTextExport` of the `CryptoKeySlotToPortPrototypeMapping` from which the object was loaded is configured as True.⌋

7.3.6 Protected Key Management

7.3.6.1 Protected Key Import

Key material can be imported into a CryptoProvider in protected form including associated meta-data such as allowed-usage. This combination of key-material and meta-data is typically generated in a trusted environment of the OEM or another authorized entity.

These update packages fall into two categories: OEM custom packages and standardized packages specified for example by SHE (e.g. as used by command SHE/LOADKEY).

FC Crypto supports both categories by specifying the ProtectedKeyImportCtx. As with other contexts the ProtectedKeyImportCtx allows a CryptoProvider to be tailored to easily fulfill project requirements. Creating a ProtectedKeyImportCtx using the string-based ara::crypto::AlgId the application developer can selected the import protocol required for his project and supported by the chosen CryptoProvider.

One example, as already mentioned above, is the data format specified in AUTOSAR_TR_SecureHardwareExtensions. A package compatible with the command SHE/LOADKEY includes a header specifying the key-slot to update and the key-slot, whose content shall be used for unwrapping the protected payload (key-encryption-key or KEK). In this use case the AlgId "SHE/LOADKEY" can be used to identify the import protocol including protection wrapper. To support SHE+ implementations, i.e. implementations with more than 10 key-slots (usually referred as banks of 10 key-slots each), the function ara::crypto::cryp::ProtectedKeyImportCtx::SetKey or ara::crypto::cryp::ProtectedKeyImportCtx::SetKey may be used. The SHE mandated response (M4, M5) can then be queried by calling ara::crypto::cryp::ProtectedKeyImportCtx::GetResponse.

In order to offer customizing the protection wrapper while preserving a standardized payload structure AUTOSAR specifies a common structure encompassing the actual key-material and meta-data available for querying from ara::crypto::cryp::CryptoObjects such as allowed-usage or AlgId. An example for a valid AlgId that might be supported by a CryptoProvider could be "ARFMT/AES-GCM", where "ARFMT" specifies the structure of the payload and "AES-GCM" specifies that AES-GCM shall be used to unwrap the payload. This format is specified below.

7.3.6.2 Protected Key Export

Protected key exporting is supported by ara::crypto::cryp::CryptoProvider::ExportSecuredKey. Similar in usage to the ProtectedKeyImportCtx it allows to specify the AlgId to be used (i.e. the protocol specifying the protection wrapper and structure of the protected payload) as well as optionally the KEK, if not implicitly specified with the AlgId. This interface is offered by the CryptoProvider directly as a dedicated context would just replicate interfaces already offered by the ara::crypto::cryp::CryptoObject to be exported.

7.3.7 Import and Export Formats

The ara::crypto::FormatId is used to define data formats and encodings for importing and exporting data to and from ara::crypto.

[SWS_CRYPT_60022] Raw Format

Upstream requirements: RS_CRYPTO_02004

⌈If a function accepts a `ara::crypto::FormatId` and the `ara::crypto::FormatId` is set to `kFormatRawValueOnly`, then the function shall (de-)serialize the object in the following formats depending on the object's type:

- SymmetricKey: The serialized, raw value of the symmetric key.
- ed25519/ed448 signature: R |S (RFC 8032).
- ECDSA signature: R |S (IEEE P1363).
- RSA PSS signature: S (RFC 8017).
- RSA PKCS#1 v1.5 signature: S (RFC 8017).
- SLH-DSA signature: R |SIG_FORS |SIG_HT (IETF draft-ietf-lamps-cms-sphincs-plus)

⌋

[SWS_CRYPT_04210] DER Format

Upstream requirements: RS_CRYPTO_02004

⌈If a function accepts a `ara::crypto::FormatId` and the `ara::crypto::FormatId` is set to `kFormatDerEncoded`, then the function shall (de-)serialize the object in DER (X.690) according to the following schema depending on the object's type:

- Public Key: SubjectPublicKeyInfo (RFC 5280)
- Private Key: OneAsymmetricKey (RFC 5958)
- Signature: signatureValue (RFC 5280)

⌋

Depending on the algorithm, different specifications define the content of the ASN.1 schema elements (e.g., SubjectPublicKeyInfo). For instance, RFC 8410 specifies how ed25519 public keys are represented in a SubjectPublicKeyInfo.

[SWS_CRYPT_04211] PEM Format

Upstream requirements: RS_CRYPTO_02004

⌈If a function accepts a `ara::crypto::FormatId` and the `ara::crypto::FormatId` is set to `kFormatPemEncoded`, then the function shall (de-)serialize the object in PEM (RFC 7468) instead of DER according to the schema specified in [SWS_CRYPT_04210].⌋

[SWS_CRYPT_60021] AUTOSAR Payload Format

Upstream requirements: RS_CRYPTO_02004

⌈If a `ara::crypto::cryp::CryptoProvider` supports the AUTOSAR payload format for import and export, then the interface `ara::crypto::cryp::Crypto-`

`Provider::ExportSecuredKey` as well as the class `ara::crypto::cryp::ProtectedKeyImportCtx` shall support exporting or importing data that complies to the following ASN.1/DER encoding specification:

ARImportFormat DEFINITIONS AUTOMATIC TAGS ::= BEGIN
RestrictedUseObject ::= SEQUENCE {
arFmtVersionNumber INTEGER,
type ObjectType,
algorithmId PrintableString,
allowedUsage INTEGER,
sessionFlag BOOLEAN,
payloadEncoding ArEncoding,
payload OCTET STRING }
ObjectType ::= ENUMERATED { kSymmetricKey (1), kPrivateKey (2), kPublicKey (3), kSecretSeed (4) }
ArEncoding ::= ENUMERATED { custom (0), raw (1), der (2), pem (3) }
END
The arFmtVersionNumber of this format shall be 0x01.⌋

7.3.8 Context Creation

A CryptoProvider implements a set of contexts, each of which provides a specific cryptographic transformation. The contexts can be created using the Create* functions provided by the CryptoProvider. The cryptographic transformation is passed as an AlgId (cf. 7.1.5) when creating a context.

7.3.9 CryptoProvider Identification

A CryptoProvider may optionally support identification, e.g., according to the SHE standard.

[SWS_CRYPT_41024] Get Provider Identification

Upstream requirements: RS_CRYPTO_02305

⌈The interface `ara::crypto::cryp::CryptoProvider::GetProviderID` shall parse the optionally provided input data of parameter in and write CryptoProvider specific identification data into the provided parameter idData or return

- `kUnsupported`, if the instantiated CryptoProvider does not support identification.
- `kInsufficientCapacity`, if the provided ReadWriteMemRegion parameter idData is insufficient in size to hold the identification data.

⌋

7.4 Crypto Contexts

This section contains the functional description of the specified transformation contexts.

7.4.1 Overview

Crypto supports contexts that can be used without a key (i.e., RandomGeneratorCtx and HashFunctionCtx), as well as contexts that require a key (all other contexts). Many contexts share the same API patterns, e.g., how keys can be set for use in a context or how data can be passed and retrieved from the context. In this section, these common pattern are described and specified.

7.4.1.1 Set Key

In order to set a key, all keyed contexts provide a SetKey function that takes either a KeySlot or a CryptoObject as a parameter. Both parameter options represent different use-cases: If an application uses a persistently stored key, it can use the KeySlot of the stored key as a parameter in the SetKey function. The context will then use the key that is stored in the KeySlot at the time the cryptographic transformation is started. If an application uses a runtime key (e.g., a generated session key), it can use the CryptoObject as a parameter in the SetKey function.

7.4.1.2 Single-call API

All contexts that can be used to process payload data (i.e., MessageAuthnCodeCtx, SignerPrivateCtx, VerifierPublicCtx, DecryptorPrivateCtx, EncryptorPublicCtx, SymmetricCipherCtx, AuthCipherCtx) provide a streaming interface as well as functions for single-call processing. The single-call processing is intended to ease and optimize calculation of small data (e.g., SecOC protected PDUs).

7.4.1.3 Streaming API

A typical usage of the streaming interface is to process large amounts of data (e.g., software update). After setting the key, the application typically first calls Start(), then calls the Update() functions multiple times with the data to be processed, and calls the Finish() function to finalize the calculation. In order to determine the maximum data size that can be processed by the context in one call to Update(), all contexts that support streaming provide a function GetMaximumChunkSize(). Similar functions exists for specific values only present in certain contexts (e.g., IV size) In order to free the resources occupied by an ongoing streaming operation without finishing the operation (e.g., because a network connection has been lost), the function Cancel() can be called.

7.4.1.4 Informational API

All contexts provide the function `ara::crypto::cryp::CryptoContext::GetAlgId` to query the AlgId of the context.

7.4.1.5 Output Buffers

In order to enable the application to control memory allocation, ara::crypto uses output buffers in the streaming and single-call APIs for all data returned from ara::crypto to the application. The required size of the output buffer may be static (e.g., depending only on the used algorithm) or it may be dependent on the current state of the context. The application can query the required output buffer size using helper functions provided by every context such as GetRequiredOutputBufferSize().

7.4.1.6 Initialization Vectors

Initialization vectors (IV) are used by certain (authenticated) encryption algorithms such as AES GCM and AES CBC. The IV is typically (randomly) generated during encryption, transmitted together with the cipher text, and then used during decryption. Reusing the same IV may compromize the security properties of the encryption scheme. Therefore, some cryptographic module implementations do not allow the application to chose an IV and instead securely generate an IV (e.g., using an internal TRNG). However, some cryptographic module implementations require the IV to be provided by the application and also certain application layer protocols require explicitly setting the IV. These options are exposed in ara::crypto by providing functions for setting the IV, but also for getting the IV in case it has been generated by the cryptographic module implementation. Specifically, the SymmetricCipherCtx and the AuthenticatedCipherCtx provide overloads of their Start() functions and their single-call functions that take an IV as an input parameter. Additionally, the SymmetricCipherCtx and the AuthenticatedCipherCtx provide overloads of their Finish() functions and their single-call functions that take an IV as an output parameter. The functions GetMinimumInputIvSize, GetMaximumInputIvSize, and GetOutputIvSize indicate to the user of ara::crypto, which variants may be used.

7.4.2 Random Number Generator (RNG)

Generating randomness or pseudo randomness is required for many operations such as creating `Salts` or `Nonces`. In order to enable applications to perform these operations, `CryptoAPI` provides an interface to generate random data.

Randomness can be generated by True `Random Number Generators` (TRNGs) or by Cryptographically Secure Pseudo `Random Number Generators` (CSPRNGs). CSPRNGs hold an internal state that needs to be securely seeded with sufficient entropy. This entropy is used to generate a deterministic but unpredictable stream of

random data. More information on the desired properties of CSPRNGs can be found in [9, BSIDRNG: Functionality Classes and Evaluation Methodology for Deterministic Random Number Generators].

Each ara::crypto::cryp::RandomGeneratorCtx may either rely on state local to the ara::crypto::cryp::RandomGeneratorCtx instance only, or may rely on global state shared among different ara::crypto::cryp::RandomGeneratorCtx's instances. In order to prevent malicious applications from being able to predict random data generated for other processes, it is important to ensure that applications must not modify the global state of any ara::crypto::cryp::RandomGeneratorCtx.

[SWS_CRYPT_00502]

Upstream requirements: RS_CRYPTO_02206

⌈If a ara::crypto::cryp::RandomGeneratorCtx uses global state, calls to its methods ara::crypto::cryp::RandomGeneratorCtx::Seed and ara::crypto::cryp::RandomGeneratorCtx::Seed shall return kUnsupported without modifying the global state.⌋

[SWS_CRYPT_40990]

Upstream requirements: RS_CRYPTO_02206

⌈The function ara::crypto::cryp::RandomGeneratorCtx::AddEntropy shall use the provided data as additional entropy (i.e., additional to the existing entropy used by the context) for random number generation.⌋

How global-state ara::crypto::cryp::RandomGeneratorCtxs are seeded is stack-vendor and/or project specific and out of scope of this specification.

A ara::crypto::cryp::RandomGeneratorCtx may have insufficient entropy to serve a request for random data, e.g., because it has not been seeded or because it ran out of entropy. In these cases, ara::crypto::cryp::RandomGeneratorCtx::Generate return either kBusyResource or kUninitializedContext. These errors represent the possible handling of the error by applications: For a global-state ara::crypto::cryp::RandomGeneratorCtx the application receives kBusyResource and has to wait, whereas for a local-state ara::crypto::cryp::RandomGeneratorCtx the application receives kUninitializedContext and has to provide additional entropy.

In addition to the ara::crypto::cryp::RandomGeneratorCtx, a CryptoProvider provides the convenience function ara::crypto::cryp::CryptoProvider::GenerateRandomData to generate random data without the need to create a separate context.

7.4.3 Hashing

A hash function is a one-way function that maps an arbitrary string of bits to a fixed-length string of bits. The main security property of a hash function is collision resistance, meaning that is computationally infeasible to construct two different inputs to the hash function, which result in the same output.

An application can use the `ara::crypto::cryp::HashFunctionCtx` for hashing.

7.4.3.1 Streaming API

The functions used for a streaming hash calculation are `ara::crypto::cryp::HashFunctionCtx::Start`, `ara::crypto::cryp::HashFunctionCtx::Start`, `ara::crypto::cryp::HashFunctionCtx::Update` and `ara::crypto::cryp::HashFunctionCtx::Update`, and `ara::crypto::cryp::HashFunctionCtx::Finish`

The function `ara::crypto::cryp::HashFunctionCtx::Start` and `ara::crypto::cryp::HashFunctionCtx::Start` may be called at any time, also if a hash calculation is currently being performed. In this case the `ara::crypto::cryp::HashFunctionCtx` will reset the internal state and start a new hash value calculation. Depending on the hash algorithm (e.g., cSHAKE (NIST SP 800 185)), the function `ara::crypto::cryp::HashFunctionCtx::Start` may accept an additional seed parameter.

The function `ara::crypto::cryp::HashFunctionCtx::Update` may be called multiple times within one hash calculation, each time providing a new chunk of data. The maximum chunk size can be queried using `ara::crypto::cryp::HashFunctionCtx::GetMaximumChunkSize`.

With the function `ara::crypto::cryp::HashFunctionCtx::Finish`, the application or `Functional Cluster` can complete the hash calculation.

Finally the function `ara::crypto::cryp::HashFunctionCtx::GetDigest` can be used to retrieve the calculated hash digest. The size of the digest can be queried using `ara::crypto::cryp::HashFunctionCtx::GetDigestSize`. The function `ara::crypto::cryp::HashFunctionCtx::Compare` can be used to compare the calculated hash digest against an expected value.

7.4.3.2 Single-call API

In addition to the streaming interface, the function `ara::crypto::cryp::HashFunctionCtx::CalculateHashDigest` provides a single-call mechanism to perform hash calculations. This mechanism can be used for efficient hash calculation of smaller data.

7.4.3.3 Suspend and Resume Hash Calculations

In order to support suspending and resuming hash calculations (e.g., across ECU restarts for hashing software update packages), the HashFunctionCtx supports exporting and importing its internal state. The function `ara::crypto::cryp::HashFunctionCtx::ExportContextState` can be used by an application to get a serialized representation of the context's internal state. An application may persist this state and provide the state at a later point in time to the function `ara::crypto::cryp::HashFunctionCtx::ImportContextState`. This will restore the state of the HashFunctionCtx to the state when the context state has been exported. Subsequently, the hash calculation can be continued. Note: It is the responsibility of the application to security persist the exported state. Tampering of the exported state may lead to incorrect hash calculations.

7.4.4 Message Authentication Code (MAC)

Message Authentication Codes (MAC) protect the integrity and authenticity of data using symmetric key cryptography. Examples for MAC algorithms are HMACs, CMACs, and GMACs.

An application can use the `ara::crypto::cryp::MessageAuthnCodeCtx` for the calculation of message authentication codes.

Before a MAC calculation can be performed, a symmetric key has to be set using `ara::crypto::cryp::MessageAuthnCodeCtx::SetKey` or a KeySlot has to be set using `ara::crypto::cryp::MessageAuthnCodeCtx::SetKey`.

The MessageAuthnCodeCtx provides a streaming API and functions for single-call MAC processing.

7.4.4.1 Streaming API

When using the streaming API, the context needs to be initialized using the function `ara::crypto::cryp::MessageAuthnCodeCtx::Start` or its overload `ara::crypto::cryp::MessageAuthnCodeCtx::Start`, which allows to set an initialization vector, which is used by some algorithms.

Data to be processed can be passed using the `ara::crypto::cryp::MessageAuthnCodeCtx::Update` function as a ReadOnlyMemRegion. This function can be called multiple times during one MAC calculation.

When using the streaming API, the MAC calculation has to be finalized by a call to `ara::crypto::cryp::MessageAuthnCodeCtx::Finish`.

After finalization of a MAC calculation, the MAC digest can be retrieved through a call to `ara::crypto::cryp::MessageAuthnCodeCtx::GetDigest`.

After finalization of a MAC calculation, the MAC digest can be compared to an expected value using the `ara::crypto::cryp::MessageAuthnCodeCtx::Compare` function or a subspan can be compared using the `ara::crypto::cryp::MessageAuthnCodeCtx::CompareSubspan` function.

7.4.4.2 Single-call API

When using the single-call API, the function `ara::crypto::cryp::MessageAuthnCodeCtx::Generate` is used to calculate and return the MAC value for data. The function `ara::crypto::cryp::MessageAuthnCodeCtx::GenerateSubspan` additionally provides a way to truncate the calculated MAC value.

The function `ara::crypto::cryp::MessageAuthnCodeCtx::Verify` is used to calculate a MAC value and compare it against an expected value. Similar to the function `ara::crypto::cryp::MessageAuthnCodeCtx::CompareSubspan`, `ara::crypto::cryp::MessageAuthnCodeCtx::VerifySubspan` can be used to calculate a MAC value and compare only a subspan of it against an expected value.

7.4.5 Digital signatures

Digital signatures are a mechanism to protect and verify the authenticity of data. A private key is used to generate a signature and the corresponding public key is used to verify the signature and thus the authenticity of the data. An application can use the `ara::crypto::cryp::SignerPrivateCtx` for creating signatures, and the `ara::crypto::cryp::VerifierPublicCtx` for verifying signatures. Both contexts provide a streaming interface and functions for single-call processing.

Signature schemes typically contain a cryptographic hash calculation as part of the overall scheme. This hash calculation can be separate from the signature calculation (e.g., ECDSA, ed25519ph), or it can be part of the signature scheme itself (e.g., ed25519). A context may either implement a signature scheme with the hashing part included, or it can implement only the signature calculation and leaves the hashing to the application. The AlgId of the context indicates whether hashing is part of the context, or a pre-hashed input is expected. For instance, the AlgId "SIG/ECDSA/secp256r1/SHA2-256" indicates that the context includes the hash calculation using SHA2-256, whereas the AlgId "SIG/ECDSA/secp256r1/PRE-HASHED" indicates that the context expects the message to be already hashed by the application. If the hashing is part of the signature scheme itself (e.g., ed25519), pre-hashing is not possible. Therefore, an AlgId "SIG/ed25519/PRE-HASHED" is not possible.

For signature verification, the public key has to be set using `ara::crypto::cryp::VerifierPublicCtx::SetKey` or a KeySlot has to be set using `ara::crypto::cryp::VerifierPublicCtx::SetKey`. For signature generation, the private key has to be set using `ara::crypto::cryp::SignerPrivateCtx::SetKey` or a KeySlot has to be set using `ara::crypto::cryp::SignerPrivateCtx::SetKey`.

7.4.5.1 Streaming API

A streaming signature generation or verification is initialized using the Start() functions. The function provides an overload that allows the application to set a "context", which is used in some signature schemes (e.g., ed25519ctx). The data to be signed is passed using the Update() function. The function GetMaximumChunkSize provides the maximum chunk size that can be passed to the Update() function. Finally, by the signature generation can be finished using the function Finish(), which writes the generated signature into an application-provided buffer. The size of the generated signature is provided by the function GetSignatureSize().

7.4.5.2 Single-call API

The function `ara::crypto::cryp::SignerPrivateCtx::Sign` can be used to create a signature in a single call. The function `ara::crypto::cryp::VerifierPublicCtx::Verify` can be used to verify a signature in a single call.

7.4.6 Symmetric Encryption & Decryption

Encryption is a mechanism to protect the confidentiality of data. Symmetric encryption uses the same symmetric key for encryption and decryption. Encryption alone does not protect the authenticity of data, however dedicated authenticated encryption schemes exist, which are described in Section 7.4.8.

The SymmetricCipherCtx supports encryption and decryption with symmetric algorithms. If the transformation includes padding, the AlgId also includes the padding performed by the context (e.g., "SYMCIPHER/AES-128/CBC/PKCS7"). If no padding is applied by the context and instead the calling application has to perform approriate padding, the AlgId ends with "PRE-PADDED" (e.g., "SYMCIPHER/AES-128/PKCS7PRE-PADDED").

Before an encryption or decryption can be performed, a SymmetricKey has to be set using `ara::crypto::cryp::SymmetricCipherCtx::SetKey` or a KeySlot has to be set using `ara::crypto::cryp::SymmetricCipherCtx::SetKey`. If a KeySlot is set via `ara::crypto::cryp::SymmetricCipherCtx::SetKey`, the Keyslot needs to contain a SymmetricKey of respective type at the time the encryption/decryption is started.

The SymmetricCipherCtx provides the following helper functions: GetRequiredOutputBufferSize to get the required size of the output buffer for the processed data. GetRecommendedIvSize to get the recommended size of the initialization vector. GetMinimumChunkSize to get the minumum number of bytes that have to be passed in a single call. GetMaximumChunkSize to get the maximum number of bytes that have to be passed in a single call.

7.4.6.1 Streaming API

When using the streaming API, the context needs to be initialized using the function `ara::crypto::cryp::SymmetricCipherCtx::Start` or its overload `ara::crypto::cryp::SymmetricCipherCtx::Start`, which allows to set an initialization vector, which is used by some algorithms. Additionally, the application can set the transformation direction, i.e., whether the context is used for encryption or decryption using the Start() function. If Start() is called during an ongoing process of the context, it will cancel the previous transformation, discard the intermediate result, and re-initialize the context for the new transformation.

Data to be processed can be passed using the `ara::crypto::cryp::SymmetricCipherCtx::Update` function. The result of the transformation is written into an application provided output buffer. The Update function can be called multiple times during one en-/decryption.

Some operation modes of specific `Stream Ciphers` are seekable, e.g., [10, CTR] or [11, Salsa20]. Seekable means that the user can efficiently seek to any position in the data stream and continue en-/decryption at that position. The function `ara::crypto::cryp::SymmetricCipherCtx::Seek` may be called in between calls to `ara::crypto::cryp::SymmetricCipherCtx::Update` to update the position in the data stream so that subsequent calls to `ara::crypto::cryp::SymmetricCipherCtx::Update` will use the set position.

The streaming processing can be finalized by a call to `ara::crypto::cryp::SymmetricCipherCtx::Finish`.

7.4.6.2 Single-call API

The function `ara::crypto::cryp::SymmetricCipherCtx::ProcessSingleCall` is used to encrypt or decrypt data in a single function call, without an IV. The function `ara::crypto::cryp::SymmetricCipherCtx::ProcessSingleCallWithInputIv` is used to encrypt or decrypt data in a single function call, with the IV being provided by the application. The function `ara::crypto::cryp::SymmetricCipherCtx::EncryptSingleCallWithOutputIv` is used to encrypt or decrypt data in a single function call, with the IV being returned by ara::crypto to the application. If any of the single-call functions is called while a streaming operation is ongoing, the streaming operation will be reset.

7.4.7 Asymmetric Encryption & Decryption

Asymmetric encryption is provided via the class EncryptorPublicCtx. Asymmetric decryption is provided via the class DecryptorPrivateCtx.

The interfaces follow the same pattern as described in Section 7.4.6. The main difference are the used key types, i.e., a PublicKey has to be used for encryption, and a PrivateKey has to be used for decryption.

7.4.8 Authenticated Encryption with Associated Data

Authenticated Encryption with Associated Data (AEAD) provide confidentiality and data authenticity simultaneously. Furthermore, authenticity can also be provided for unencrypted plaintext data ("authenticated data"). This supports use-cases like IPsec or MACsec where the header of a network frame must be transmitted in plaintext but its authenticity is still protected, and only the payload of the network frame is additionally encrypted.

The AuthCipherCtx supports AEAD. Its interface is similar to the SymmetricCipherCtx with the additional functions and parameters dealing with the additional authenticated data:

UpdateAuthenticatedData() can be called after Start and before confidential data has been processed in order to pass the additional authenticate data. The Finish() function has an additional parameter for the authentication tag. For decryption, the tag is provided by the application and the function verifies the authenticity of the processed data using the tag. For encryption, the tag is written to the provided output buffer.

Note: During decryption, the function UpdateConfidentialData will return plaintext data without verifying its authenticity. An application must not use the data before verifying its authenticity using Finish(). For instance, an application may use the streaming API to store the decrypted data on an inactive partition before verifying its authenticity, however, it must not set the partition to active before verifying the authenticity of the data.

The ProcessSingleCall() function also has an additional parameter for the authenticated data and for the authentication tag. Similar to the Finish() function, the tag is either used for verification during decryption, or to write the generated authentication tag during encryption.

Additional helper functions support operation with additional data and the buffer for the tag: GetTagSize to get the required size of the output buffer to which the authentication tag is written to. GetMaxAssociatedDataSize to get the maximum length of the associated data supported by the context.

7.4.9 Key Derivation Function (KDF)

A key-derivation function can be used to derive additional keys from an already established or generated key in a way that knowledge of a derived key does not leak information on other derived keys or the key used as input the the KDF.

The KeyDerivationFunctionCtx provides key derivation.

Before a key derivation can be performed, a symmetric key has to be set using `ara::crypto::cryp::KeyDerivationFunctionCtx::SetKey` or a KeySlot has to be set using `ara::crypto::cryp::KeyDerivationFunctionCtx::SetKey`. A key that is used for key derivation must have the AllowedUsage kAllowKeyDerivation.

All standardized KDFs support to parameterize the deriviation with one or more parameters. Because the parameter identification differs between algorithms, each parameter is identified as a String according to Table 7.2. Each parameters can be set via the SetParameter() function either as an input buffer, a SymmetricKey, or as a uint64_t.

Parameter Name	*Specification*	*Value Type*
`Salt`	NIST SP 800-56C, HKDF, PBKDF	Buffer, SymmetricKey
`FixedInfo`	NIST SP 800-56C	Buffer
`SharedInfo`	ANSI X9.63	Buffer
`Info`	HKDF	Buffer
`Label`	NIST SP 800-108r1, TLS PRF	Buffer
`Context`	NIST SP 800-108r1	Buffer
`IterationCount`	PBKDF	Integer
`IV`	NIST SP 800-108r1	Buffer
`Seed`	TLS PRF	Buffer

Table 7.2: Standardized Parameters

If a KDF uses a parameter not listed in the table, a CryptoProvider implementation may extend the accepted parameters to configure the KDF.

After setting all required paramaters, the function `ara::crypto::cryp::KeyDerivationFunctionCtx::DeriveKey` can be called to derive a new key. The application can set the AlgId of the to be derived key (e.g., "SYMKEY/AES-128") and has to specify the AllowedUsage of the derived key.

7.4.10 Key Wrapping

Key Wrapping (as defined in [12] and [13]) encapsulates `Key Material`, which is used for example to store a key in an unsecure environment or transport a key by an unsecure channel. Wrapping a key is a kind of encryption of the key and contributes to confidentiality.

The SymmetricKeyWrapperCtx provides key wrapping.

Wrapping a key requires a key encryption key (`KEK`), which can be set via the functions `ara::crypto::cryp::SymmetricKeyWrapperCtx::SetKey` to set a SymmetricKey or `ara::crypto::cryp::SymmetricKeyWrapperCtx::SetKey` to configure a KeySlot. The allowed usage of the KEK must include kAllowKeyWrapping for wrapping and kAllowKeyUnwrapping for unwrapping.

The function ara::crypto::cryp::SymmetricKeyWrapperCtx::Wrap can be used to wrap key material provided by the parameter object. The allowed usage of the to be wrapped key must include kAllowSecureExport. The size of the serialized wrapped key can be queried via GetRequiredWrappedKeyBufferSize in order to allocate an apropriately sized output buffer.

The function ara::crypto::cryp::SymmetricKeyWrapperCtx::UnwrapSymmetricKey can be used to unwrap a symmetric key. Similar to key generation, the application has to set the AlgId and the allowed usage of the unwrapped key.

7.4.11 Key Encapsulation Mechanism (KEM)

Key encapsulation mechanisms (KEM) are used to protect the confidentiality of a symmetric key that is exchanged using public key cryptography. The sender of the symmetric key uses the recipients public key for encryption, and the receiver uses its private key to decrypt the shared symmetric key.

The KeyEncapsulatorPublicCtx provides key encapsulation, the KeyDecapsulatorPrivateCtx provides key decapsulation.

For both contexts, the public/private key has to be set first via SetKey. The allowed usage of this key must include kAllowKeyWrapping for encapsulation and kAllowKeyUnwrapping for decapsulation.

Depending on the algorithm, the symmetric key is either passed as an input parameter, or generated as part of the encapsulation. Therefore, the function ara::crypto::cryp::KeyEncapsulatorPublicCtx::Encapsulate provides two overloads, one to pass a SymmetricKey to be enapsulated, and one that generates and returns a new SymmetricKey.

The function ara::crypto::cryp::KeyDecapsulatorPrivateCtx::Decapsulate performs decapsulation of a key. Similar to key generation, the application has to set the AlgId and the allowed usage of the unwrapped key.

7.4.12 Key Agreement

Key Agreement schemes like Diffie-Hellman [14] allow two parties to agree on a shared secret without revealing information about the shared secret to third parties. The shared secret is typically used as an input for a key derivation function, which derives the subsequently used symmetric keys. The KeyAgreementPrivateCtx provides an interface for key agreement.

Before a key agreement can be performed, a private key matching the key agreement protocol has to be generated. Key agreement requires as input the public key of the communication partner (other side). To retrieve an instance of ara::crypto::cryp::PublicKey representing the public key received from the communication partner, the interface ara::crypto::cryp::CryptoProvider::ImportPublicKey

can be used. Similarly, the communication partner requires the public key of the local application. To send this public data the interface `ara::crypto::Serializable::ExportPublicly` can be used to retrieve the raw data of the public key. Each `ara::crypto::cryp::PublicKey` instance provides this interface. The public key of the other party is used as input to the function AgreeKey to calculate the agreed secret.

While the agreed secret can be used as input to a KeyDerivationFunctionCtx, some HSM implementations (e.g., PKCS#11) execute the key derivation together with the key agreement. In such a case, the AlgId of the context represents the additional KDF. For instance, a "KEYAGREEMENT/ECDH/secp256r1" represents a context performing an elliptic curve Diffie-Hellman key exachange on curve secp256r1, while "KEYAGREEMENT/ECDH/secp256r1/ANSI_X9_63/SHA2-256" additionally performs a KDF according to ANSI X9.63 with SHA2-256. The KDF can be configured using the SetParameter functions and the AgreeKey function additionally executes the KDF identified by the AlgId.

7.5 Certificate handling (X.509 Provider)

The X509Provider is responsible for:

- Parsing and validating X.509 certificates and certificate chains.
- Persisting and loading certificates.
- Creating Certificate Signing Requests
- Parsing, validating and storing CRLs.
- Creating OCSP requests and parsing and validating responses.

The X.509 Provider is instantiated using the function `ara::crypto::LoadX509Provider`.

7.5.1 X.509 Certificate Parsing

Applications can parse individual certificates using the function `ara::crypto::x509::X509Provider::ParseCert`. A complete certification chain (e.g., as received during a TLS handshake) can be parsed into a Vector of Certificate objects using the function `ara::crypto::x509::X509Provider::ParseCertChain`.

The returned `ara::crypto::x509::Certificate` object represents the parsed certificate. It provides the following functions for getting attributes of the certificate:

- `ara::crypto::x509::Certificate::GetCertificateSignatureAlgorithm`
- `ara::crypto::x509::Certificate::GetSerialNumber`
- `ara::crypto::x509::Certificate::GetIssuer`

- ara::crypto::x509::Certificate::GetNotBefore
- ara::crypto::x509::Certificate::GetNotAfter
- ara::crypto::x509::Certificate::GetSubject
- ara::crypto::x509::Certificate::GetSubjectPublicKeyAlgorithm
- ara::crypto::x509::Certificate::LoadSubjectPublicKey
- ara::crypto::x509::Certificate::GetAuthorityKeyIdentifierExtension
- ara::crypto::x509::Certificate::GetSubjectKeyIdentifierExtension
- ara::crypto::x509::Certificate::GetKeyUsageExtension
- ara::crypto::x509::Certificate::GetExtendedKeyUsageExtension

Most functions are getter functions that return the requested attribute. The function ara::crypto::x509::Certificate::LoadSubjectPublicKey instantiates the subject public key in a CryptoProvider chosen by the application via the parameter cryptoProvider and sets the allowed usage according to the certificate's key usage extension.

[SWS_CRYPT_20907]

Upstream requirements: RS_CRYPTO_02306

⌈The function ara::crypto::x509::Certificate::LoadSubjectPublicKey shall instantiate the certificate's subject public key in the provided cryptoProvider. If the certificate does not contain the key usage extension (RFC 5280, Section 4.2.1.3), all allowed usage flags applicable to PublicKeys shall be set (i.e., kAllowSigVerification |kAllowKeyWrapping |kAllowDataEncryption |kAllowKeyAgreement |kAllowPlaintextExport |kAllowPersist). If the certificate contains the key usage extension, AllowExport and kAllowPersist shall be set. Additionally the following key usage flags shall be set for each certificate "key usage":

- digitalSignature: kAllowSigVerification
- nonRepudiation: kAllowSigVerification
- keyEncipherment: kAllowKeyWrapping
- dataEncipherment: kAllowDataEncryption
- keyAgreement: kAllowKeyAgreement
- keyCertSign: kAllowSigVerification
- cRLSign: kAllowSigVerification

- encipherOnly: kAllowKeyWrapping |kAllowDataEncryption
- decipherOnly: kAllowKeyUnwrapping |kAllowDataDecryption

⌋

Distinguished Names are represented by the class `ara::crypto::x509::X509DN`. A DN object can be converted to a string representation using the function `ara::crypto::x509::X509DN::ToString` or can be serialized using the function `ara::crypto::Serializable::ExportPublicly`. The key usages of the subject public key according to the key usage extension and the extended key usage extensions are represented as bit fields of types `ara::crypto::x509::X509KeyUsage` and `ara::crypto::x509::X509ExtendedKeyUsage`, respectively.

7.5.2 X.509 Certificate Validation

Validation of certificate chains can be performed using the `ara::crypto::x509::CertificateValidationCtx`. An application can create a `ara::crypto::x509::CertificateValidationCtx` by calling the function `ara::crypto::x509::X509Provider::CreateCertificateValidationCtx`. An application then typically configures the `ara::crypto::x509::CertificateValidationCtx` using its functions.

The function `ara::crypto::x509::CertificateValidationCtx::AddTrustedCertificate` allows an application to set a certificate as a trusted certificate. This function may be called multiple times in order to add multiple trusted certificates (e.g., representing different potential trust anchors for a TLS connection).

The function `ara::crypto::x509::CertificateValidationCtx::AddUntrustedCertificate` allows an application to set a certificate as an untrusted certificate. Such a certificate will be used by the path validation as a potential intermediate certificate that links a trusted certificate to the target certificate. This function may be called multiple times in order to add multiple untrusted certificates (e.g., representing different potential intermediate certificates used for diagnostic access control).

In order to verify the notBefore and notAfter attributes of a certificate, the current time can be set with the function `ara::crypto::x509::CertificateValidationCtx::SetCurrentTime`. If this function is not called, the system time will be used during certificate validation.

According to RFC 5280, a certificate-using system must reject the certificate if it encounters a critical extension it does not recognize or a critical extension that contains information that it cannot process. In order to support critical extensions that are validated by the application instead of the `ara::crypto::x509::CertificateValidationCtx`, the application can call the function `ara::crypto::x509::`

CertificateValidationCtx::IgnoreCriticalExtension and add those extension OIDs that the application validates. The certificate path validation performed by ara::crypto will then accept a certificate, even it contains the added critical extensions.

While a complete validation according to RFC 5280 is the default case, there can be scenarios where certain aspects of certificate validation need to be omitted for operational or legacy purposed. The function ara::crypto::x509::CertificateValidationCtx::SetValidationOptions enables the application to enable or disable individual aspects of certificate path validation.

In order to add one or more OCSP responses to be considered during certificate validation, the function ara::crypto::x509::CertificateValidationCtx::AddOcspResponse can be called by an application. The function validated the OCSP using the current configuration of the ara::crypto::x509::CertificateValidationCtx, e.g., using the trusted certificates. When using OCSP, the application additionally needs to decide whether only the leaf certificate or all certificates in the chain need to have an OCSP response by setting kOcspResponseTargetOnly or kOcspResponseCompleteChain, respectively, via the function ara::crypto::x509::CertificateValidationCtx::SetValidationOptions.

The certificate path validation is done using the functions ara::crypto::x509::CertificateValidationCtx::ValidateCertificate or ara::crypto::x509::CertificateValidationCtx::ValidateCertificateChain.

The function ara::crypto::x509::CertificateValidationCtx::ValidateCertificate takes only the target certificate and the ara::crypto::x509::CertificateValidationCtx establishes a certificate path based on the context configuration (e.g., the trusted and untrusted certificates). The function ara::crypto::x509::CertificateValidationCtx::ValidateCertificateChain takes a complete certificate chain as input and validates the provided chain against the context configuration.

The result of a certificate path validation is represented by the class ara::crypto::x509::CertificateValidationResult. The function ara::crypto::x509::CertificateValidationResult::GetValidationStatus provides the overall status of the certificate path validation. The function overload ara::crypto::x509::CertificateValidationResult::GetValidationStatus can be used to query the status of individual certificates of the identified certificate chain. The function ara::crypto::x509::CertificateValidationResult::GetCertificateChain can be used to obtain the complete certificate chain that has been found during certificate path validation. An application might use this, e.g., for logging.

7.5.3 X.509 Certificate Storage

The X509Provider provides two ways to store certificates: Certificates can be stored individually, i.e., with one modelled CryptoCertificate representing the potential storage location of one certificate. Furthermore, certificates can be stored in certificate

groups, i.e., with one modelled `CryptoCertificateGroup` representing the potential storage location of multiple certificates. Certificate groups provide a means to deal with a larger number of certificates that are typically handled in a similar way by the application. For example, an application may have a large number of potential trusted root certificates for establishing a backend connection. All such root certificates could be stored in one certificate group and used by an application during certificate path validation.

The function `ara::crypto::x509::X509Provider::ImportCertificate` persists a certificate using the provided InstanceSpecifier as an identifier for the storage location. The function `ara::crypto::x509::X509Provider::LoadCertificate` loads a previously stored certificate from persistent memory. The function `ara::crypto::x509::X509Provider::ClearCertificateSlot` deletes a previously stored certificate from persistent memory.

The function `ara::crypto::x509::X509Provider::ImportCertificateInGroup` persists an additional certificate in a group. Already persisted certificates in the group are not modified. The function `ara::crypto::x509::X509Provider::LoadCertificateGroup` loads all certificates stored in the group and referenced by the group for usage by the application. The function `ara::crypto::x509::X509Provider::RemoveCertificateFromGroup` removes an individual certificate from a certificate group. Only certificate stored in the CertificateGroup storage itself can be removed using this functions. Certificates stored in a `CryptoCertificate` referenced by `CryptoCertificateGroup` in the role `includedCertificate` can not be deleted using the function RemoveCertificateFromGroup but must be deleted using the function `ara::crypto::x509::X509Provider::ClearCertificateSlot`.

All access to persisted certificates is subject to access control (see [SWS_CRYPT_60002] and [SWS_CRYPT_60003]).

[SWS_CRYPT_20618] Protected Certificate Storage

Upstream requirements: RS_CRYPTO_02306

⌈`Crypto` shall protect the stored certificates from unauthorized access and tampering.⌋

[SWS_CRYPT_20620] Protected Certificate Storage

Upstream requirements: RS_CRYPTO_02306

⌈`Crypto` shall persist certificates in the location identified by the parameter `uri` of the `ArtifactLocator` that references the `CryptoCertificate` or `CryptoCertificateGroup` in the role `representedModelElement`.⌋

Note: [constr_10380] prevents two ArtifactLocators pointing to the same `CryptoCertificate` or `CryptoCertificateGroup`.

Note: If there is no ArtifactLocator that references the CryptoCertificate or CryptoCertificateGroup in the role representedModelElement, then the storage location is not specified by AUTOSAR and stack-vendor specific.

7.5.4 Certificate Signing Requests (CSR)

Crypto supports the creation of certificate signing requests using the function ara::crypto::x509::X509Provider::CreateCertSignRequest. The function takes a Distinguished Name (DNs) as an input parameter. DNs can be created using the functions ara::crypto::x509::X509Provider::CreateDn (from a string representation) and ara::crypto::x509::X509Provider::CreateDn (from a serialized buffer).

7.5.5 Certificate Revocation Lists (CRL)

The function ara::crypto::x509::X509Provider::ImportCrl imports and persists a CRL. The function takes an InstanceSpecifier that is mapped to a CryptoCertificate, storing the CA certificate of the CRL.

The function ara::crypto::x509::X509Provider::DeleteCrl deletes a previously stored CRL.

7.5.6 Online Certificate Status Protocol (OCSP)

The X509Provider supports creating OCSP requests and parsing OCSP responses.

OCSP requests can be created using the functions ara::crypto::x509::X509Provider::CreateOcspRequest and ara::crypto::x509::X509Provider::CreateOcspRequest. The functions return the created OCSP request as a serialized buffer. The application can then transmit the OCSP request to an OCSP responder in order to receive an OCSP response. The function ara::crypto::x509::X509Provider::ParseOcspResponse can then be used for parsing the OCSP response and it returns a ara::crypto::x509::OcspResponse representing the parsed OCSP response. The function ara::crypto::x509::OcspResponse::GetResponseStatus can be used to query the status of the response. The function ara::crypto::x509::OcspResponse::GetCertificates allows an application to get certificates embedded in the OCSP response. The OCSP response can be used during certificate path validation (cf. ara::crypto::x509::CertificateValidationCtx::AddOcspResponse).

[SWS_CRYPT_41019] OCSP Request Generation

Upstream requirements: RS_CRYPTO_02306

⌈The `CryptoAPI` provides methods `ara::crypto::x509::X509Provider::CreateOcspRequest`, `ara::crypto::x509::X509Provider::CreateOcspRequest` to generate an `ara::crypto::x509::OcspRequest` request, which is defined in [15, RFC 6960]. The method can be used by an application or `Functional Cluster`.⌋

7.6 Reporting

7.6.1 Security Events

This section lists all security events defined by this functional cluster.

[SWS_CRYPT_41022] Security events for Crypto

Status: DRAFT

Upstream requirements: RS_Ids_00810

⌈

Name	*Description*	*ID*
SEV_CERT_CHAIN_VERIFICATION_FAILED	The verification of a certificate against a certificate chain was not successful.	5
SEV_CERT_INSTALL	Attempt to install a new certificate.	95
SEV_CERT_UPDATE	Attempt to update a certificate.	96
SEV_CERT_DELETE	Attempt to delete a certificate.	97
SEV_ACCESS_CONTROL_CRYPTO_IAM_ACCESS_DENIED	Access of an application to a resource provided by Cryptography was denied.	134

⌋

[SWS_CRYPT_41023] Security event context data definition: SEV_ACCESS_CONTROL_CRYPTO_IAM_ACCESS_DENIED

Status: DRAFT

Upstream requirements: RS_CRYPTO_02004

⌈

SEV Name	*SEV_ACCESS_CONTROL_CRYPTO_IAM_ACCESS_DENIED*	
ID	134	
Description	Access of an application to a resource provided by Cryptography was denied.	
Context Data Version	1	
Context Data	*Data Type*	*Allowed Values*
UserId	uint32	

⌋

[SWS_CRYPT_41021] Access Control Security Event

Upstream requirements: RS_CRYPTO_02004

⌈If access to resources was not granted by `FC Crypto` due to [SWS_CRYPT_10000], [SWS_CRYPT_10001], [SWS_CRYPT_60000], [SWS_CRYPT_60001], [SWS_CRYPT_60002], [SWS_CRYPT_60003] `FC Crypto` shall raise the `SEv SEV_ACCESS_CONTROL_CRYPTO_IAM_ACCESS_DENIED` to the `IdsM`.⌋

[SWS_CRYPT_41059] Security event context data definition: SEV_CERT_CHAIN_VERIFICATION_FAILED

Upstream requirements: RS_Ids_00810

⌈

SEV Name	***SEV_CERT_CHAIN_VERIFICATION_FAILED***	
ID	5	
Description	The verification of a certificate against a certificate chain was not successful.	
Context Data Version	2	
Context Data	***Data Type***	***Allowed Values***
CertError	uint8	Cert_no_error (0x00) Cert_invalid_format (0x01) Cert_invalid_type (0x02) Cert_invalid_chain_of_trust (0x03) Cert_signature_fail (0x04) Cert_revoked (0x05) Cert_validity_period_fail (0x06) Cert_invalid_content (0x07) General_failure (0xFF)
Length	uint16	Length of Certificate byte array
Certificate	uint8 [100]	Certificate, optionally truncated to the first KeyMSEv CertificateMaxLength bytes
	MAX-LENGTH	Truncation length can be defined on a project specific basis. AUTOSAR has defined a default value, as given in the Certificate byte array definition above.

⌋

[SWS_CRYPT_41064] Reporting verification of a certificate chain failed

Upstream requirements: RS_Ids_00810

⌈If a verification of a certificate against a certificate chain was not successful, `FC Crypto` shall raise the `SEv SEV_CERT_CHAIN_VERIFICATION_FAILED` to the `IdsM`.⌋

[SWS_CRYPT_41060] Security event context data definition: SEV_CERT_INSTALL

Upstream requirements: RS_Ids_00810

⌈

SEV Name	***SEV_CERT_INSTALL***	
ID	95	
Description	Attempt to install a new certificate.	
Context Data Version	1	
Context Data	***Data Type***	***Allowed Values***
CertError	uint8	Cert_no_error (0x00) Cert_invalid_format (0x01) Cert_invalid_type (0x02) Cert_invalid_chain_of_trust (0x03) Cert_signature_fail (0x04) Cert_revoked (0x05) Cert_validity_period_fail (0x06) Cert_invalid_content (0x07) General_failure (0xFF)
Length	uint16	Length of Certificate byte array
Certificate	uint8 [100]	Certificate, optionally truncated to the first KeyMSEv CertificateMaxLength bytes
	MAX-LENGTH	Truncation length can be defined on a project specific basis. AUTOSAR has defined a default value, as given in the Certificate byte array definition above.

⌋

[SWS_CRYPT_41063] Reporting initial persistent certificate installation

Upstream requirements: RS_Ids_00810

⌈If a certificate is persisted in an empty `CryptoCertificate` or a `CryptoCertificateGroup`, `FC Crypto` shall raise the `SEv SEV_CERT_INSTALL` to the `IdsM`.⌋

[SWS_CRYPT_41061] Security event context data definition: SEV_CERT_UPDATE

Upstream requirements: RS_Ids_00810

⌈

SEV Name	***SEV_CERT_UPDATE***	
ID	96	
Description	Attempt to update a certificate.	
Context Data Version	1	
Context Data	***Data Type***	***Allowed Values***
CertError	uint8	Cert_no_error (0x00) Cert_invalid_format (0x01) Cert_invalid_type (0x02) Cert_invalid_chain_of_trust (0x03) Cert_signature_fail (0x04) Cert_revoked (0x05) Cert_validity_period_fail (0x06) Cert_invalid_content (0x07) General_failure (0xFF)

▽

△

SEV Name	*SEV_CERT_UPDATE*	
Length	uint16	Length of Certificate byte array
Certificate	uint8 [100]	Certificate, optionally truncated to the first KeyMSEv CertificateMaxLength bytes
	MAX-LENGTH	Truncation length can be defined on a project specific basis. AUTOSAR has defined a default value, as given in the Certificate byte array definition above.

⌋

[SWS_CRYPT_41065] Reporting updated persistent certificate installation

Upstream requirements: RS_Ids_00810

⌈If a certificate is persisted in a `CryptoCertificate` that contained a certificate to be replaced, `FC Crypto` shall raise the `SEv SEV_CERT_UPDATE` to the `IdsM`.⌋

[SWS_CRYPT_41062] Security event context data definition: SEV_CERT_DELETE

Upstream requirements: RS_Ids_00810

⌈

SEV Name	*SEV_CERT_DELETE*	
ID	97	
Description	Attempt to delete a certificate.	
Context Data Version	1	
Context Data	*Data Type*	*Allowed Values*
CertError	uint8	Cert_no_error (0x00) Cert_invalid_format (0x01) Cert_invalid_type (0x02) Cert_invalid_chain_of_trust (0x03) Cert_signature_fail (0x04) Cert_revoked (0x05) Cert_validity_period_fail (0x06) Cert_invalid_content (0x07) General_failure (0xFF)
Length	uint16	Length of Certificate byte array
Certificate	uint8 [100]	Certificate, optionally truncated to the first KeyMSEv CertificateMaxLength bytes
	MAX-LENGTH	Truncation length can be defined on a project specific basis. AUTOSAR has defined a default value, as given in the Certificate byte array definition above.

⌋

[SWS_CRYPT_41066] Reporting removal of a persistent certificate

Upstream requirements: RS_Ids_00810

⌈If a certificate is removed from persistent storage in a `CryptoCertificate` or a `CryptoCertificateGroup`, `FC Crypto` shall raise the `SEv SEV_CERT_DELETE` to the `IdsM`.⌋

Please note that the certificate truncation use-case from CP KeyM is not supported by the FC Crypto. The FC Crypto will always provide the complete certificate to the IdsM.

7.6.2 Log Messages

This section lists all non-verbose log messages (i.e., modelled DLT messages) defined by this functional cluster.

[SWS_CRYPT_41043] LogMessage InitializationFailed

Status: DRAFT
Upstream requirements: RS_CRYPTO_02110

⌈

Dlt-Message	InitializationFailed		
Description	When an application calls ara::core::Initialize, while the Crypto FC can't correctly initialize the structure for this application		
MessageId	0x80002001		
MessageType Info	DLT_LOG_ERROR		
Dlt-Argument	***ArgumentDescription***	***ArgumentType***	***ArgumentUnit***
modeledProcessId	Meta-model identifier of the process that caused the violation, i.e., its short name path with '/' as a separator	uint32	NoUnit

⌋

[SWS_CRYPT_41044] LogMessage DeinitializationFailed

Status: DRAFT
Upstream requirements: RS_CRYPTO_02110

⌈

Dlt-Message	DeinitializationFailed		
Description	When an application calls ara::core::DeInitialize, while the Crypto FC can't correctly DeInitialize the structure for this application		
MessageId	0x80002002		
MessageType Info	DLT_LOG_ERROR		
Dlt-Argument	***ArgumentDescription***	***ArgumentType***	***ArgumentUnit***
modeledProcessId	Meta-model identifier of the process that caused the violation, i.e., its short name path with '/' as a separator	uint32	NoUnit

⌋

[SWS_CRYPT_41045] LogMessage CryptoProviderLoadingFailed

Status: DRAFT
Upstream requirements: RS_CRYPTO_02110

⌈

Dlt-Message	CryptoProviderLoadingFailed		
Description	When a CryptoProvider can't be loaded		
MessageId	0x80002003		
MessageType Info	DLT_LOG_ERROR		
Dlt-Argument	***ArgumentDescription***	***ArgumentType***	***ArgumentUnit***
cryptoProvider InstanceSpecifier	An instance specifier for the CryptoProvider that failed to load	uint8 [encoding UTF-8]	

⌋

[SWS_CRYPT_41047] LogMessage ResourceAccessNotGranted

Status: DRAFT
Upstream requirements: RS_CRYPTO_02110

⌈

Dlt-Message	ResourceAccessNotGranted		
Description	When an application attempts to reach a specific resource but IAM don't grant the access because the application doesn't have an access to the resource		
MessageId	0x80002005		
MessageType Info	DLT_LOG_WARN		
Dlt-Argument	***ArgumentDescription***	***ArgumentType***	***ArgumentUnit***
modeledProcess Id	Meta-model identifier of the process that caused the violation, i.e., its short name path with '/' as a separator	uint32	NoUnit
resourceInstance Specifier	The identifier for the resource that IAM denied access to. This could refer to either a provider or a key slot instance specifier	uint8 [encoding UTF-8]	

⌋

[SWS_CRYPT_41049] LogMessage KeySlotContentChanged

Status: DRAFT
Upstream requirements: RS_CRYPTO_02110

⌈

Dlt-Message	KeySlotContentChanged		
Description	When, by any means, a KeySlot content is changed, either added, deleted, or modified. Crypto SHALL log the event as info		
MessageId	0x80002007		
MessageType Info	DLT_LOG_INFO		
Dlt-Argument	***ArgumentDescription***	***ArgumentType***	***ArgumentUnit***

▽

△

keySlotInstance Specifier	An instance specifier for the KeySlot whose contents have been changed indicates which KeySlot is affected	uint8 [encoding UTF-8]	

⌋

7.6.3 Violation Messages

This section lists all violation messages (i.e., DLT messages logged for Violations according to [SWS_CORE_00021]) defined by this functional cluster.
Please note that concrete implementations might additionally implement Non-Standardized Violations (see also [SWS_CORE_00003]).

[SWS_CORE_13005]

Dlt-Message	ProcessMappingViolation		
Description	Matching InstanceRef exists, but no matching (modelled) Process found that matches the (runtime) process.		
MessageId	0x80001ffa		
MessageType Info	DLT_LOG_FATAL		
Dlt-Argument	***ArgumentDescription***	***ArgumentType***	***ArgumentUnit***
modeledProcessId	Meta-model identifier of the process that caused the violation, i.e., its short name path with '/' as a separator.	uint8 [encoding UTF-8]	
location	An implementation-defined identifier of the location where the violation was detected, for example {filename}:{linenumber}.	uint8 [encoding UTF-8]	
instanceSpecifier	InstanceSpecifier used to try to create the object.	uint8 [encoding UTF-8]	
className	Name of the class that was instantiated.	uint8 [encoding UTF-8]	

[SWS_CORE_13000]

Dlt-Message	InsufficientPermissionsViolation		
Description	Sent in case the caller had insufficient permissions for the requested operation.		
MessageId	0x80001fff		
MessageType Info	DLT_LOG_FATAL		
Dlt-Argument	***ArgumentDescription***	***ArgumentType***	***ArgumentUnit***
modeledProcessId	Meta-model identifier of the process that caused the violation, i.e., its short name path with '/' as a separator.	uint8 [encoding UTF-8]	
location	An implementation-defined identifier of the location where the violation was detected, for example {filename}:{linenumber}.	uint8 [encoding UTF-8]	
message	Additional message that describes the cause of the access violation.	uint8 [encoding UTF-8]	

7.6.4 Production Errors

This functional cluster does not define any production errors (i.e., Diagnostic Events).

8 API specification

This chapter provides a reference of the APIs defined by this functional cluster. The API is described in the following chapters in tables. Table 8.1 explains the content that is described in such an API table.

Kind:	Defines the kind of the declaration that this API table describes. The following values are supported: • class (Declaration of a class) • function (Declaration of a member or non-member function) • struct (Declaration of a structure) • type alias (Declaration of a type alias) • enumeration (Declaration of an enumeration) • variable (Declaration of a variable)	
Port Interfaces:	States that the C++ API `class` is the related C++ API binding for the given modeled sub-class of `PortInterface`	
Header File:	Defines the header file to be included according to [SWS_CORE_90001]	
Forwarding Header File:	Defines the forwarding header file to be included according to [SWS_CORE_90001]	
Scope:	Defines the scope that may be a C++ `namespace` (in case of a `class` or non-member function) or a `class` declaration (in case of a member)	
Symbol:	C++ symbol name	
Thread Safety:	Defines whether a function is thread-safe, not thread-safe, or conditional according to [SWS_CORE_13200] and [SWS_CORE_13202]	
Syntax:	Description of C++ syntax	
Template Param:	Template parameter (0..*)	Template parameter(s) used to parameterize the template
Parameters (in):	Parameter declaration (0..*)	Parameter(s) that are passed to the function
Parameters (out):	Parameter declaration (0..*)	Parameter(s) that are returned to the caller
Return Value:	Return type	Type of the value that the function returns
Exception Safety:	Defines whether a function is exception-safe, not exception safe or conditionally exception safe	
Exceptions:	List of C++ `Exception`s that may be thrown by the function	
Violations:	List of violations that may raised by the function	
Errors:	Error type (0..*)	List of defined `ara::core::ErrorCode`s that may be returned by the function with their recoverability class defined in [RS_AP_00160]. APIs can be extended with vendor-specific error codes. These are not standardized by AUTOSAR
Description:	Brief description of the function	

Table 8.1: Explanation of an API table

8.1 PortInterface to API class binding

This table shows the `API class` binding for each `PortInterface` owned by this functional cluster and those functions taking an `ara::core::InstanceSpecifier` argument, designated to "construct" that class. These constructing functions may be any combination of special-member constructors, named constructor members or non-member factory constructors.

Port Interface	API Class / Function
CryptoCertificateGroupInterface	[SWS_CRYPT_40600] Definition of API class ara::crypto::x509::X509Provider
	[SWS_CRYPT_50126] Definition of API function ara::crypto::x509::X509Provider::LoadCertificateGroup
CryptoCertificateInterface	[SWS_CRYPT_40600] Definition of API class ara::crypto::x509::X509Provider
	[SWS_CRYPT_40641] Definition of API function ara::crypto::x509::X509Provider::LoadCertificate
CryptoKeySlotClientInterface	[SWS_CRYPT_50145] Definition of API class ara::crypto::cryp::KeySlot
	[SWS_CRYPT_50004] Definition of API function ara::crypto::LoadKeySlot
CryptoKeySlotInterface	[SWS_CRYPT_50166] Definition of API class ara::crypto::cryp::WriteableKeySlot
	[SWS_CRYPT_50005] Definition of API function ara::crypto::LoadWriteableKeySlot
CryptoProviderInterface	[SWS_CRYPT_20700] Definition of API class ara::crypto::cryp::CryptoProvider
	[SWS_CRYPT_20099] Definition of API function ara::crypto::LoadCryptoProvider

Table 8.2: PortInterface (sub-class) to API class / function binding

8.2 Header: ara/crypto/common/base_id_types.h

8.2.1 Non-Member Types

8.2.1.1 Type Alias: AlgId

[SWS_CRYPT_10014] Definition of API type ara::crypto::AlgId

Upstream requirements: RS_CRYPTO_02102, RS_CRYPTO_02107

⌈

Kind:	type alias
Header file:	#include "ara/crypto/common/base_id_types.h"
Scope:	namespace ara::crypto
Symbol:	AlgId
Syntax:	`using AlgId = ara::core::StringView;`
Description:	Type for holding StringView of algorithm according to standardized format described in the chapter 'Naming Convention'.

⌋

8.2.1.2 Type Alias: AlgIdString

[SWS_CRYPT_50000] Definition of API type ara::crypto::AlgIdString

Upstream requirements: RS_CRYPTO_02102, RS_CRYPTO_02107

⌈

Kind:	type alias
Header file:	#include "ara/crypto/common/base_id_types.h"
Scope:	namespace ara::crypto
Symbol:	AlgIdString
Syntax:	`using AlgIdString = ara::core::String;`
Description:	Type for holding String of algorithm according to standardized format described in the chapter 'Naming Convention'.

⌋

8.2.1.3 Type Alias: AllowedUsageFlags

[SWS_CRYPT_10015] Definition of API type ara::crypto::AllowedUsageFlags

Upstream requirements: RS_CRYPTO_02111

⌈

Kind:	type alias
Header file:	#include "ara/crypto/common/base_id_types.h"
Scope:	namespace ara::crypto
Symbol:	AllowedUsageFlags
Syntax:	`using AllowedUsageFlags = std::uint32_t;`
Description:	Type for holding bit-flags of allowed usages of a key.

⌋

8.2.1.4 Enumeration: CryptoObjectType

[SWS_CRYPT_10016] Definition of API enum ara::crypto::CryptoObjectType

Upstream requirements: RS_CRYPTO_02004

⌈

Kind:	enumeration
Header file:	#include "ara/crypto/common/base_id_types.h"
Forwarding header file:	#include "ara/crypto/crypto_fwd.h"
Scope:	namespace ara::crypto
Symbol:	CryptoObjectType
Underlying type:	std::uint8_t

▽

△

Syntax:	`enum class CryptoObjectType : std::uint8_t {...};`	
Values:	kSymmetricKey	= 1
		cryp::SymmetricKey object
	kPrivateKey	= 2
		cryp::PrivateKey object
	kPublicKey	= 3
		cryp::PublicKey object
Description:	Enumeration of all types of crypto objects, i.e. types of content that can be stored to a key slot.	

⌋

8.2.1.5 Enumeration: CryptoTransform

[SWS_CRYPT_10019] Definition of API enum ara::crypto::CryptoTransform

Upstream requirements: RS_CRYPTO_02004

⌈

Kind:	enumeration	
Header file:	#include "ara/crypto/common/base_id_types.h"	
Forwarding header file:	#include "ara/crypto/crypto_fwd.h"	
Scope:	namespace ara::crypto	
Symbol:	CryptoTransform	
Underlying type:	std::uint8_t	
Syntax:	`enum class CryptoTransform : std::uint8_t {...};`	
Values:	kEncrypt	= 1
		encryption
	kDecrypt	= 2
		decryption
Description:	Enumeration of cryptographic transformation directions.	

⌋

8.2.1.6 Type Alias: Timestamp

[SWS_CRYPT_50003] Definition of API type ara::crypto::Timestamp

Upstream requirements: RS_CRYPTO_02306

⌈

Kind:	type alias
Header file:	#include "ara/crypto/common/base_id_types.h"
Scope:	namespace ara::crypto
Symbol:	Timestamp

▽

△

Syntax:	`using Timestamp = std::chrono::time_point<std::chrono::system_clock, std::chrono::seconds>;`
Description:	Type for timestamps.

⌋

8.2.2 Global Variables

8.2.2.1 kAllowDataDecryption

[SWS_CRYPT_13102] Definition of API variable ara::crypto::kAllowDataDecryption

Upstream requirements: RS_CRYPTO_02111

⌈

Kind:	variable
Header file:	#include "ara/crypto/common/base_id_types.h"
Scope:	namespace ara::crypto
Symbol:	kAllowDataDecryption
Type:	AllowedUsageFlags
Syntax:	`constexpr AllowedUsageFlags kAllowDataDecryption {0x0002};`
Description:	The key can be used for data decryption.

⌋

8.2.2.2 kAllowDataEncryption

[SWS_CRYPT_13101] Definition of API variable ara::crypto::kAllowDataEncryption

Upstream requirements: RS_CRYPTO_02111

⌈

Kind:	variable
Header file:	#include "ara/crypto/common/base_id_types.h"
Scope:	namespace ara::crypto
Symbol:	kAllowDataEncryption
Type:	AllowedUsageFlags
Syntax:	`constexpr AllowedUsageFlags kAllowDataEncryption {0x0001};`
Description:	The following constants define 1-bit values representing allowed usages. The key can be used for data encryption

⌋

8.2.2.3 kAllowDerivedPlaintextExport

[SWS_CRYPT_13150] Definition of API variable ara::crypto::kAllowDerivedPlaintextExport

Upstream requirements: RS_CRYPTO_02111

⌈

Kind:	variable
Header file:	#include "ara/crypto/common/base_id_types.h"
Scope:	namespace ara::crypto
Symbol:	kAllowDerivedPlaintextExport
Type:	AllowedUsageFlags
Syntax:	`constexpr AllowedUsageFlags kAllowDerivedPlaintextExport {0x4000};`
Description:	The derived key can be exported in plaintext.

⌋

8.2.2.4 kAllowHashCalculation

[SWS_CRYPT_13110] Definition of API variable ara::crypto::kAllowHashCalculation

Upstream requirements: RS_CRYPTO_02111

⌈

Kind:	variable
Header file:	#include "ara/crypto/common/base_id_types.h"
Scope:	namespace ara::crypto
Symbol:	kAllowHashCalculation
Type:	AllowedUsageFlags
Syntax:	`constexpr AllowedUsageFlags kAllowHashCalculation {0x1000};`
Description:	The key can be used as input to a HashFunctionCtx.

⌋

8.2.2.5 kAllowKeyAgreement

[SWS_CRYPT_13105] Definition of API variable ara::crypto::kAllowKeyAgreement

Upstream requirements: RS_CRYPTO_02111

⌈

Kind:	variable
Header file:	#include "ara/crypto/common/base_id_types.h"
Scope:	namespace ara::crypto
Symbol:	kAllowKeyAgreement
Type:	AllowedUsageFlags
Syntax:	`constexpr AllowedUsageFlags kAllowKeyAgreement {0x0020};`
Description:	The key can be used as private or public input to key agreement.

⌋

8.2.2.6 kAllowKeyDerivation

[SWS_CRYPT_13106] Definition of API variable ara::crypto::kAllowKeyDerivation

Upstream requirements: RS_CRYPTO_02111

⌈

Kind:	variable
Header file:	#include "ara/crypto/common/base_id_types.h"
Scope:	namespace ara::crypto
Symbol:	kAllowKeyDerivation
Type:	AllowedUsageFlags
Syntax:	`constexpr AllowedUsageFlags kAllowKeyDerivation {0x0040};`
Description:	The key can be used for key derivation.

⌋

8.2.2.7 kAllowKeyUnwrapping

[SWS_CRYPT_50002] Definition of API variable ara::crypto::kAllowKeyUnwrapping

Upstream requirements: RS_CRYPTO_02111

⌈

Kind:	variable
Header file:	#include "ara/crypto/common/base_id_types.h"
Scope:	namespace ara::crypto

▽

△

Symbol:	kAllowKeyUnwrapping
Type:	AllowedUsageFlags
Syntax:	`constexpr AllowedUsageFlags kAllowKeyUnwrapping {0x0100};`
Description:	The key can be used as a key encryption key for key unwrapping or key decapsulation.

⌋

8.2.2.8 kAllowKeyWrapping

[SWS_CRYPT_13108] Definition of API variable ara::crypto::kAllowKeyWrapping

Upstream requirements: RS_CRYPTO_02111

⌈

Kind:	variable
Header file:	#include "ara/crypto/common/base_id_types.h"
Scope:	namespace ara::crypto
Symbol:	kAllowKeyWrapping
Type:	AllowedUsageFlags
Syntax:	`constexpr AllowedUsageFlags kAllowKeyWrapping {0x0080};`
Description:	The key can be used as a key encryption key for key wrapping or key encapsulation.

⌋

8.2.2.9 kAllowMac

[SWS_CRYPT_50001] Definition of API variable ara::crypto::kAllowMac

Upstream requirements: RS_CRYPTO_02111

⌈

Kind:	variable
Header file:	#include "ara/crypto/common/base_id_types.h"
Scope:	namespace ara::crypto
Symbol:	kAllowMac
Type:	AllowedUsageFlags
Syntax:	`constexpr AllowedUsageFlags kAllowMac {0x0004};`
Description:	The key can be used for MAC generation and verification.

⌋

8.2.2.10 kAllowPersist

[SWS_CRYPT_13109] Definition of API variable ara::crypto::kAllowPersist

Upstream requirements: RS_CRYPTO_02111

⌈

Kind:	variable
Header file:	#include "ara/crypto/common/base_id_types.h"
Scope:	namespace ara::crypto
Symbol:	kAllowPersist
Type:	AllowedUsageFlags
Syntax:	`constexpr AllowedUsageFlags kAllowPersist {0x0200};`
Description:	The key can persisted.

⌋

8.2.2.11 kAllowPlaintextExport

[SWS_CRYPT_40992] Definition of API variable ara::crypto::kAllowPlaintextExport

Upstream requirements: RS_CRYPTO_02111

⌈

Kind:	variable
Header file:	#include "ara/crypto/common/base_id_types.h"
Scope:	namespace ara::crypto
Symbol:	kAllowPlaintextExport
Type:	AllowedUsageFlags
Syntax:	`constexpr AllowedUsageFlags kAllowPlaintextExport {0x2000};`
Description:	The key can be exported in plaintext.

⌋

8.2.2.12 kAllowRngInit

[SWS_CRYPT_13107] Definition of API variable ara::crypto::kAllowRngInit

Upstream requirements: RS_CRYPTO_02111

⌈

Kind:	variable
Header file:	#include "ara/crypto/common/base_id_types.h"
Scope:	namespace ara::crypto
Symbol:	kAllowRngInit

▽

△

Type:	AllowedUsageFlags
Syntax:	`constexpr AllowedUsageFlags kAllowRngInit {0x0800};`
Description:	The symmetric key can be used for seeding of a RandomGeneratorCtx.

⌋

8.2.2.13 kAllowSecuredExport

[SWS_CRYPT_40991] Definition of API variable ara::crypto::kAllowSecuredExport

Upstream requirements: RS_CRYPTO_02111

⌈

Kind:	variable
Header file:	#include "ara/crypto/common/base_id_types.h"
Scope:	namespace ara::crypto
Symbol:	kAllowSecuredExport
Type:	AllowedUsageFlags
Syntax:	`constexpr AllowedUsageFlags kAllowSecuredExport {0x0400};`
Description:	The key can be securely exported (protected export)

⌋

8.2.2.14 kAllowSigGeneration

[SWS_CRYPT_13103] Definition of API variable ara::crypto::kAllowSigGeneration

Upstream requirements: RS_CRYPTO_02111

⌈

Kind:	variable
Header file:	#include "ara/crypto/common/base_id_types.h"
Scope:	namespace ara::crypto
Symbol:	kAllowSigGeneration
Type:	AllowedUsageFlags
Syntax:	`constexpr AllowedUsageFlags kAllowSigGeneration {0x0008};`
Description:	The key can be used for signature generation.

⌋

8.2.2.15 kAllowSigVerification

[SWS_CRYPT_13104] Definition of API variable ara::crypto::kAllowSigVerification

Upstream requirements: RS_CRYPTO_02111

⌈

Kind:	variable
Header file:	#include "ara/crypto/common/base_id_types.h"
Scope:	namespace ara::crypto
Symbol:	kAllowSigVerification
Type:	AllowedUsageFlags
Syntax:	`constexpr AllowedUsageFlags kAllowSigVerification {0x0010};`
Description:	The key can be used for signature verification.

⌋

8.2.3 Struct: SecureCounter

[SWS_CRYPT_30001] Definition of API class ara::crypto::SecureCounter

Upstream requirements: RS_CRYPTO_02401

⌈

Kind:	struct
Header file:	#include "ara/crypto/common/base_id_types.h"
Forwarding header file:	#include "ara/crypto/crypto_fwd.h"
Scope:	namespace ara::crypto
Symbol:	SecureCounter
Syntax:	`struct SecureCounter {...};`
Description:	128 bit secure counter made up of two uint64_t's representing the most significant and least significant bits

⌋

8.2.3.1 Public Member Variables

8.2.3.1.1 mLSQW

[SWS_CRYPT_30002] Definition of API variable ara::crypto::SecureCounter::mLSQW

Upstream requirements: RS_CRYPTO_02401

⌈

Kind:	variable
Header file:	#include "ara/crypto/common/base_id_types.h"
Scope:	`struct ara::crypto::SecureCounter`
Symbol:	mLSQW
Type:	std::uint64_t
Syntax:	`std::uint64_t mLSQW;`
Description:	least significant 64 bits

⌋

8.2.3.1.2 mMSQW

[SWS_CRYPT_30003] Definition of API variable ara::crypto::SecureCounter::mMSQW

Upstream requirements: RS_CRYPTO_02401

⌈

Kind:	variable
Header file:	#include "ara/crypto/common/base_id_types.h"
Scope:	`struct ara::crypto::SecureCounter`
Symbol:	mMSQW
Type:	std::uint64_t
Syntax:	`std::uint64_t mMSQW;`
Description:	most significant 64 bits

⌋

8.3 Header: ara/crypto/common/crypto_error_domain.h

8.3.1 Non-Member Types

8.3.1.1 Enumeration: CryptoErrc

[SWS_CRYPT_10099] Definition of API enum ara::crypto::CryptoErrc

Upstream requirements: RS_CRYPTO_02310

⌈

Kind:	enumeration	
Header file:	#include "ara/crypto/common/crypto_error_domain.h"	
Forwarding header file:	#include "ara/crypto/crypto_fwd.h"	
Scope:	namespace ara::crypto	
Symbol:	CryptoErrc	
Underlying type:	ara::core::ErrorDomain::CodeType	
Syntax:	`enum class CryptoErrc : ara::core::ErrorDomain::CodeType {...};`	
Values:	kBusyResource	= 16777217U
		ResourceException: Specified resource is busy!
	kNoSuchValue	= 2U
		kNoSuchElement: The requested element is not present in the object.
	kInvalidArgument	= 33619968U
		InvalidArgumentException: An invalid argument value is provided!
	kUnknownIdentifier	= 33619969U
		InvalidArgumentException: Unknown identifier is provided!
	kInsufficientCapacity	= 33619970U
		InvalidArgumentException: Insufficient capacity of the output buffer!
	kInvalidInputSize	= 33619971U
		InvalidArgumentException: Invalid size of an input buffer!
	kIncompatibleArguments	= 33619972U
		InvalidArgumentException: Provided values of arguments are incompatible!
	kOutOfBounds	= 67174407U
		InvalidArgumentException: Provided value is above the upper boundary!
	kAuthTagNotValid	= 33619976U
		AuthTagNotValidException: Provided authentication-tag cannot be verified!
	kUnsupported	= 33620224U
		UnsupportedException: Unsupported request (due to limitations of the implementation)!
	kInvalidUsageOrder	= 33685504U
		InvalidUsageOrderException: Invalid usage order of the interface!
	kUninitializedContext	= 33685505U
		InvalidUsageOrderException: Context of the interface was not initialized!

▽

△

	kProcessingNotStarted	= 33685506U
		InvalidUsageOrderException: Data processing was not started yet!
	kProcessingNotFinished	= 33685507U
		InvalidUsageOrderException: Data processing was not finished yet!
	kUnsupportedFormat	= 50331649U
		RuntimeException: Unsupported serialization format for this object type!
	kIncompatibleObject	= 50397185U
		UnexpectedValueException: The provided object is incompatible with requested operation or its configuration!
	kEmptyKeySlot	= 50397187U
		UnexpectedValueException: Specified container is empty!
	kUsageViolation	= 50462720U
		UsageViolationException: Violation of allowed usage for the object!
	kEmptyCertificateSlot	= 50462721U
		--
Description:	Defines the error codes for the `ara::crypto::CryptoErrorDomain`	

⌋

8.3.2 Non-Member Functions

8.3.2.1 Other

8.3.2.1.1 GetCryptoErrorDomain

[SWS_CRYPT_19952] Definition of API function ara::crypto::GetCryptoErrorDomain

Upstream requirements: RS_CRYPTO_02310

⌈

Kind:	function	
Header file:	#include "ara/crypto/common/crypto_error_domain.h"	
Scope:	namespace ara::crypto	
Syntax:	`constexpr const ara::core::ErrorDomain & GetCryptoErrorDomain () noexcept;`	
Return value:	const ara::core::Error Domain &	the CryptoErrorDomain
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Return a reference to the global CryptoErrorDomain.	

⌋

8.3.2.1.2 MakeErrorCode

[SWS_CRYPT_19951] Definition of API function ara::crypto::MakeErrorCode

Upstream requirements: RS_CRYPTO_02310

⌈

Kind:	function	
Header file:	#include "ara/crypto/common/crypto_error_domain.h"	
Scope:	namespace ara::crypto	
Syntax:	`constexpr ara::core::ErrorCode MakeErrorCode (CryptoErrorDomain::Errc code, ara::core::ErrorDomain::SupportDataType data) noexcept;`	
Parameters (in):	code	an error code identifier from the `CryptoErrc` enumeration
	data	supplementary data for the error description
Return value:	ara::core::ErrorCode	an instance of `ErrorCode` created according the arguments
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Makes Error Code instances from the Crypto Error Domain. The returned ErrorCode instance always references to CryptoErrorDomain.	

⌋

8.3.3 Class: CryptoErrorDomain

[SWS_CRYPT_19900] Definition of API class ara::crypto::CryptoErrorDomain

Upstream requirements: RS_AP_00130

⌈

Kind:	class
Header file:	#include "ara/crypto/common/crypto_error_domain.h"
Forwarding header file:	#include "ara/crypto/crypto_fwd.h"
Scope:	namespace ara::crypto
Symbol:	CryptoErrorDomain
Base class:	ara::core::ErrorDomain
Syntax:	`class CryptoErrorDomain final :  public ara::core::ErrorDomain {...};`
Unique ID:	As per `ara::crypto::CryptoErrorDomain` in [SWS_CORE_90023]
Description:	Crypto Error Domain class that provides interfaces as defined by ara::core::ErrorDomain such as a name of the Crypto Error Domain or messages for each error code. This class represents an error domain responsible for all errors that may be reported by public APIs in ara::crypto namespace. .

⌋

8.3.3.1 Public Member Types

8.3.3.1.1 Type Alias: Errc

[SWS_CRYPT_19903] Definition of API type ara::crypto::CryptoErrorDomain::Errc

Upstream requirements: RS_CRYPTO_02310

⌈

Kind:	type alias
Header file:	#include "ara/crypto/common/crypto_error_domain.h"
Scope:	`class ara::crypto::CryptoErrorDomain`
Symbol:	Errc
Syntax:	`using Errc = CryptoErrc;`
Description:	crypto error

⌋

8.3.3.1.2 Type Alias: Exception

[SWS_CRYPT_19904] Definition of API type ara::crypto::CryptoErrorDomain::Exception

Upstream requirements: RS_CRYPTO_02310

⌈

Kind:	type alias
Header file:	#include "ara/crypto/common/crypto_error_domain.h"
Scope:	`class ara::crypto::CryptoErrorDomain`
Symbol:	Exception
Syntax:	`using Exception = CryptoException;`
Description:	Alias for the exception base class.

⌋

8.3.3.2 Public Member Functions

8.3.3.2.1 Special Member Functions

8.3.3.2.1.1 Default Constructor

[SWS_CRYPT_19902] Definition of API function ara::crypto::CryptoErrorDomain::CryptoErrorDomain

Upstream requirements: RS_CRYPTO_02310

⌈

Kind:	function
Header file:	#include "ara/crypto/common/crypto_error_domain.h"
Scope:	`class ara::crypto::CryptoErrorDomain`
Syntax:	`constexpr CryptoErrorDomain () noexcept;`
Exception Safety:	exception safe
Thread Safety:	thread-safe
Description:	Ctor of the CryptoErrorDomain .

⌋

8.3.3.2.2 Member Functions

8.3.3.2.2.1 Message

[SWS_CRYPT_19953] Definition of API function ara::crypto::CryptoErrorDomain::Message

Upstream requirements: RS_CRYPTO_02310

⌈

Kind:	function	
Header file:	#include "ara/crypto/common/crypto_error_domain.h"	
Scope:	`class ara::crypto::CryptoErrorDomain`	
Syntax:	`const char * Message (ara::core::ErrorDomain::CodeType errorCode) const noexcept override;`	
Parameters (in):	errorCode	an error code identifier from the `CryptoErrc` enumeration
Return value:	const char *	message text of error code
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Translate an error code value into a text message.	

⌋

8.3.3.2.2.2 Name

[SWS_CRYPT_19950] Definition of API function ara::crypto::CryptoErrorDomain::Name

Upstream requirements: RS_CRYPTO_02310

⌈

Kind:	function	
Header file:	#include "ara/crypto/common/crypto_error_domain.h"	
Scope:	`class ara::crypto::CryptoErrorDomain`	
Syntax:	`const char * Name () const noexcept override;`	
Return value:	const char *	As per `ara::crypto::CryptoErrorDomain` in [SWS_CORE_90023]
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Returns the unique name of the error domain.	

⌋

8.3.3.2.2.3 ThrowAsException

[SWS_CRYPT_19954] Definition of API function ara::crypto::CryptoErrorDomain::ThrowAsException

Upstream requirements: RS_CRYPTO_02310

⌈

Kind:	function	
Header file:	#include "ara/crypto/common/crypto_error_domain.h"	
Scope:	`class ara::crypto::CryptoErrorDomain`	
Syntax:	`[[noreturn]] void ThrowAsException (const ara::core::ErrorCode &error Code) const override;`	
Parameters (in):	errorCode	an error code identifier from the `CryptoErrc` enumeration
Return value:	None	
Exception Safety:	not exception safe	
Thread Safety:	thread-safe	
Description:	throws exception of error code. As per [SWS_CORE_10304], this function does not participate in overload resolution when C++ exceptions are disabled in the compiler toolchain.	

⌋

8.3.4 Class: CryptoException

[SWS_CRYPT_19905] Definition of API class ara::crypto::CryptoException

Upstream requirements: RS_CRYPTO_02310

⌈

Kind:	class
Header file:	#include "ara/crypto/common/crypto_error_domain.h"
Forwarding header file:	#include "ara/crypto/crypto_fwd.h"
Scope:	namespace ara::crypto
Symbol:	CryptoException
Base class:	ara::core::Exception
Syntax:	`class CryptoException : public ara::core::Exception {...};`
Description:	Exception type thrown for CRYPTO errors.

⌋

8.3.4.1 Public Member Functions

8.3.4.1.1 Constructors

8.3.4.1.1.1 CryptoException

[SWS_CRYPT_19906] Definition of API function ara::crypto::CryptoException::CryptoException

Upstream requirements: RS_CRYPTO_02310

⌈

Kind:	function	
Header file:	#include "ara/crypto/common/crypto_error_domain.h"	
Scope:	`class ara::crypto::CryptoException`	
Syntax:	`explicit CryptoException (ara::core::ErrorCode err) noexcept;`	
Parameters (in):	err	the ErrorCode
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Construct a new CryptoException from an ErrorCode.	

⌋

8.4 Header: ara/crypto/common/entry_point.h

8.4.1 Non-Member Functions

8.4.1.1 Other

8.4.1.1.1 LoadCryptoProvider

[SWS_CRYPT_20099] Definition of API function ara::crypto::LoadCryptoProvider

Upstream requirements: RS_CRYPTO_02401, RS_CRYPTO_02301

⌈

Kind:	function	
Header file:	#include "ara/crypto/common/entry_point.h"	
Scope:	namespace ara::crypto	
Syntax:	`ara::core::Result< cryp::CryptoProvider::Sptr > LoadCryptoProvider (const ara::core::InstanceSpecifier &iSpecify) noexcept;`	
Parameters (in):	iSpecify	InstanceSpecifier of the RPortPrototype of type CryptoProvider Interface
Return value:	ara::core::Result< ara::crypto::cryp::CryptoProvider::Sptr >	shared pointer to loaded Crypto Provider
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	This function does not specify any standardized errors.	
Violations:	`ProcessMappingViolation`	In case InstanceSpecifier does not point to a PortPrototype typed by a CryptoProviderInterface modeled for the current process.
Description:	This factory method shall create or return the existing instance of a specific Crypto Provider.	

⌋

8.4.1.1.2 LoadKeySlot

[SWS_CRYPT_50004] Definition of API function ara::crypto::LoadKeySlot

Upstream requirements: RS_CRYPTO_02004

⌈

Kind:	function	
Header file:	#include "ara/crypto/common/entry_point.h"	
Scope:	namespace ara::crypto	
Syntax:	`ara::core::Result< cryp::KeySlot::Sptr > LoadKeySlot (const ara::core::InstanceSpecifier &iSpecify) noexcept;`	
Parameters (in):	iSpecify	instance specifier to the RPortPrototype of a CryptoKeySlot Interface or a CryptoKeySlotClientInterface that identifies the key to be loaded
Return value:	ara::core::Result< cryp::KeySlot::Sptr >	a result containing either the KeySlot or an error
Exception Safety:	exception safe	

▽

△

Thread Safety:	thread-safe	
Errors:	This function does not specify any standardized errors.	
Violations:	`ProcessMappingViolation`	In case InstanceSpecifier does not point to a PortPrototype typed by a CryptoKeySlotInterface or CryptoKeySlotClientInterface modeled for the current process.
Description:	Load a read-only key slot. This factory method shall load the information associated with the specified KeySlot into a KeySlot object.	

⌋

8.4.1.1.3 LoadWriteableKeySlot

[SWS_CRYPT_50005] Definition of API function ara::crypto::LoadWriteableKeySlot

Upstream requirements: RS_CRYPTO_02004

⌈

Kind:	function	
Header file:	#include "ara/crypto/common/entry_point.h"	
Scope:	namespace ara::crypto	
Syntax:	`ara::core::Result< cryp::WriteableKeySlot::Sptr > LoadWriteableKeySlot (const ara::core::InstanceSpecifier &iSpecify) noexcept;`	
Parameters (in):	iSpecify	instance specifier to the RPortPrototype of a CryptoKeySlot Interface that identifies the key to be loaded
Return value:	ara::core::Result< cryp::WriteableKey Slot::Sptr >	a result containing either the KeySlot or an error
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	This function does not specify any standardized errors.	
Violations:	`ProcessMappingViolation`	In case InstanceSpecifier does not point to a PortPrototype typed by a CryptoKeySlotInterface modeled for the current process.
Description:	Load a writeable key slot. This factory method shall load the information associated with the specified KeySlot into a WriteableKeySlot object.	

⌋

8.4.1.1.4 LoadX509Provider

[SWS_CRYPT_40099] Definition of API function ara::crypto::LoadX509Provider

Upstream requirements: RS_CRYPTO_02306, RS_CRYPTO_02301

⌈

Kind:	function	
Header file:	#include "ara/crypto/common/entry_point.h"	
Scope:	namespace ara::crypto	
Syntax:	`ara::core::Result< x509::X509Provider::Sptr > LoadX509Provider () noexcept;`	
Return value:	ara::core::Result< ara::crypto::x509::X509 Provider::Sptr >	shared pointer to loaded X.509 Provider
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	This function does not specify any standardized errors.	
Description:	This factory method shall create or return the existing instance of the singleton X.509 Provider.	

⌋

8.5 Header: ara/crypto/common/formatter.h

8.5.1 Non-Member Functions

8.5.1.1 Other

8.5.1.1.1 LoadFormatter

[SWS_CRYPT_50200] Definition of API function ara::crypto::LoadFormatter

Upstream requirements: RS_CRYPTO_02112

⌈

Kind:	function	
Header file:	#include "ara/crypto/common/formatter.h"	
Scope:	namespace ara::crypto	
Syntax:	`ara::core::Result< Formatter::Sptr > LoadFormatter () noexcept;`	
Return value:	ara::core::Result< ara::crypto::Formatter::Sptr >	shared pointer to Formatter
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	This function does not specify any standardized errors.	
Description:	Factory that creates a Formatter.	

⌋

8.5.2 Class: Formatter

[SWS_CRYPT_50190] Definition of API class ara::crypto::Formatter

Upstream requirements: RS_CRYPTO_02105

⌈

Kind:	class
Header file:	#include "ara/crypto/common/formatter.h"
Forwarding header file:	#include "ara/crypto/crypto_fwd.h"
Scope:	namespace ara::crypto
Symbol:	Formatter
Syntax:	`class Formatter {...};`
Description:	Formatter object interface.

⌋

8.5.2.1 Public Member Types

8.5.2.1.1 Type Alias: Sptr

[SWS_CRYPT_50191] Definition of API type ara::crypto::Formatter::Sptr

Upstream requirements: RS_CRYPTO_02306

⌈

Kind:	type alias
Header file:	#include "ara/crypto/common/formatter.h"
Scope:	`class ara::crypto::Formatter`
Symbol:	Sptr
Syntax:	`using Sptr = std::shared_ptr<Formatter>;`
Description:	Smart pointer of the interface.

⌋

8.5.2.2 Public Member Functions

8.5.2.2.1 Special Member Functions

8.5.2.2.1.1 Copy Constructor

[SWS_CRYPT_50195] Definition of API function ara::crypto::Formatter::Formatter

Upstream requirements: RS_CRYPTO_02004

⌈

Kind:	function
Header file:	#include "ara/crypto/common/formatter.h"
Scope:	`class ara::crypto::Formatter`
Syntax:	`Formatter (const Formatter &)=delete;`
Description:	Copy-Constructor.

⌋

8.5.2.2.1.2 Move Constructor

[SWS_CRYPT_50196] Definition of API function ara::crypto::Formatter::Formatter

Upstream requirements: RS_CRYPTO_02004

⌈

Kind:	function
Header file:	#include "ara/crypto/common/formatter.h"
Scope:	`class ara::crypto::Formatter`
Syntax:	`Formatter (Formatter &&)=delete;`
Description:	Move-Constructor.

⌋

8.5.2.2.1.3 Copy Assignment Operator

[SWS_CRYPT_50193] Definition of API function ara::crypto::Formatter::operator=

Upstream requirements: RS_CRYPTO_02004

⌈

Kind:	function
Header file:	#include "ara/crypto/common/formatter.h"
Scope:	`class ara::crypto::Formatter`
Syntax:	`Formatter & operator= (const Formatter &other)=delete;`
Description:	Copy-assign another Formatter to this instance.

⌋

8.5.2.2.1.4 Move Assignment Operator

[SWS_CRYPT_50194] Definition of API function ara::crypto::Formatter::operator=

Upstream requirements: RS_CRYPTO_02004

⌈

Kind:	function
Header file:	#include "ara/crypto/common/formatter.h"
Scope:	`class ara::crypto::Formatter`
Syntax:	`Formatter & operator= (Formatter &&other)=delete;`
Description:	Move-assign another Formatter to this instance.

⌋

8.5.2.2.1.5 Destructor

[SWS_CRYPT_50192] Definition of API function ara::crypto::Formatter::~Formatter

Upstream requirements: RS_CRYPTO_02004, RS_CRYPTO_02302

⌈

Kind:	function
Header file:	#include "ara/crypto/common/formatter.h"
Scope:	`class ara::crypto::Formatter`
Syntax:	`virtual ~Formatter () noexcept=default;`
Exception Safety:	exception safe
Description:	Destructor.

⌋

8.5.2.2.2 Member Functions

8.5.2.2.2.1 Convert

[SWS_CRYPT_50198] Definition of API function ara::crypto::Formatter::Convert

Upstream requirements: RS_CRYPTO_02112

⌈

Kind:	function	
Header file:	#include "ara/crypto/common/formatter.h"	
Scope:	`class ara::crypto::Formatter`	
Syntax:	`virtual ara::core::Result< void > Convert (ReadOnlyMemRegion input, ReadWriteMemRegion out) const noexcept=0;`	
Parameters (in):	input	an input buffer containing the data in the input format
Parameters (out):	out	an output buffer that the data will be written to in the output format
Return value:	ara::core::Result< void >	An empty result or an error.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kInvalidArgument	rollback_semantics
		if `serialized` does not conform to the specified format
	ara::crypto::CryptoErrc::kInsufficientCapacity	rollback_semantics
		if the output buffer is too small
Description:	Converts the data according to the previous configuration of the Formatter.	

⌋

8.5.2.2.2.2 GetExportSize

[SWS_CRYPT_50199] Definition of API function ara::crypto::Formatter::GetExportSize

Upstream requirements: RS_CRYPTO_02112

⌈

Kind:	function	
Header file:	#include "ara/crypto/common/formatter.h"	
Scope:	`class ara::crypto::Formatter`	
Syntax:	`virtual ara::core::Result< std::size_t > GetExportSize () const noexcept=0;`	
Return value:	ara::core::Result< std::size_t >	A result containing the size in bytes of the serialized object or an error.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	This function does not specify any standardized errors.	
Description:	Get the size in bytes of the object when serialized using the current configuration of the Formatter.	

⌋

8.5.2.2.2.3 SetFormatConfiguration

[SWS_CRYPT_50197] Definition of API function ara::crypto::Formatter::SetFormatConfiguration

Upstream requirements: RS_CRYPTO_02112

⌈

Kind:	function	
Header file:	#include "ara/crypto/common/formatter.h"	
Scope:	`class ara::crypto::Formatter`	
Syntax:	`virtual ara::core::Result< void > SetFormatConfiguration (FormatId inputFormat, FormatId outputFormat, AlgId algId) noexcept=0;`	
Parameters (in):	inputFormat	the input format
	outputFormat	the output format
	algId	the AlgId of the data to be formatted
Return value:	ara::core::Result< void >	An empty result or an error.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kUnsupportedFormat	rollback_semantics
		if the specified conversion is not supported
Description:	Configure the formatter with the expected input format, output format, and the AlgId of the data to be formatted. The input and output format may be kFormatRawValueOnly, kFormatDer Encoded, or kFormatPemEncoded.	

⌋

8.6 Header: ara/crypto/common/mem_region.h

8.6.1 Non-Member Types

8.6.1.1 Type Alias: ReadOnlyMemRegion

[SWS_CRYPT_10033] Definition of API type ara::crypto::ReadOnlyMemRegion

Upstream requirements: RS_CRYPTO_02004

⌈

Kind:	type alias
Header file:	#include "ara/crypto/common/mem_region.h"
Scope:	namespace ara::crypto
Symbol:	ReadOnlyMemRegion
Syntax:	`using ReadOnlyMemRegion = ara::core::Span<const std::uint8_t>;`
Description:	Read-Only Memory Region (intended for [in] arguments)

⌋

8.6.1.2 Type Alias: ReadWriteMemRegion

[SWS_CRYPT_10031] Definition of API type ara::crypto::ReadWriteMemRegion

Upstream requirements: RS_CRYPTO_02004

⌈

Kind:	type alias
Header file:	#include "ara/crypto/common/mem_region.h"
Scope:	namespace ara::crypto
Symbol:	ReadWriteMemRegion
Syntax:	`using ReadWriteMemRegion = ara::core::Span<std::uint8_t>;`
Description:	Read-Write Memory Region (intended for [out] or [in/out] arguments)

⌋

8.7 Header: ara/crypto/common/serializable.h

8.7.1 Non-Member Types

8.7.1.1 Type Alias: FormatId

[SWS_CRYPT_10701] Definition of API type ara::crypto::FormatId

Upstream requirements: RS_CRYPTO_02004, RS_CRYPTO_02302

⌈

Kind:	type alias
Header file:	#include "ara/crypto/common/serializable.h"
Scope:	namespace ara::crypto
Symbol:	FormatId
Syntax:	`using FormatId = std::uint32_t;`
Description:	A container type for the encoding format identifiers.

⌋

8.7.2 Global Variables

8.7.2.1 kFormatDerEncoded

[SWS_CRYPT_60018] Definition of API variable ara::crypto::kFormatDerEncoded

Upstream requirements: RS_CRYPTO_02004, RS_CRYPTO_02302

⌈

Kind:	variable
Header file:	#include "ara/crypto/common/serializable.h"
Scope:	namespace ara::crypto
Symbol:	kFormatDerEncoded
Type:	FormatId
Syntax:	`constexpr FormatId kFormatDerEncoded {2};`
Description:	Export DER-encoded value of an object.

⌋

8.7.2.2 kFormatPemEncoded

[SWS_CRYPT_60019] Definition of API variable ara::crypto::kFormatPemEncoded

Upstream requirements: RS_CRYPTO_02004, RS_CRYPTO_02302

⌈

Kind:	variable
Header file:	#include "ara/crypto/common/serializable.h"
Scope:	namespace ara::crypto
Symbol:	kFormatPemEncoded
Type:	FormatId
Syntax:	`constexpr FormatId kFormatPemEncoded {3};`
Description:	Export PEM-encoded value of an object.

⌋

8.7.2.3 kFormatRawValueOnly

[SWS_CRYPT_60017] Definition of API variable ara::crypto::kFormatRawValueOnly

Upstream requirements: RS_CRYPTO_02004, RS_CRYPTO_02302

⌈

Kind:	variable
Header file:	#include "ara/crypto/common/serializable.h"
Scope:	namespace ara::crypto
Symbol:	kFormatRawValueOnly
Type:	FormatId
Syntax:	`constexpr FormatId kFormatRawValueOnly {1};`
Description:	Export only raw value of an object, i.e., a plain byte array for symmetric keys and ISO/IEC 7816-8 / IEEE P1363 encoding for combined values.

⌋

8.7.2.4 kProprietaryFormat

[SWS_CRYPT_60016] Definition of API variable ara::crypto::kProprietaryFormat

Upstream requirements: RS_CRYPTO_02004, RS_CRYPTO_02302

⌈

Kind:	variable
Header file:	#include "ara/crypto/common/serializable.h"
Scope:	namespace ara::crypto
Symbol:	kProprietaryFormat
Type:	FormatId
Syntax:	`constexpr FormatId kProprietaryFormat {0};`
Description:	Proprietary format that is only valid in the context of a CryptoProvider.

⌋

8.7.3 Class: Serializable

[SWS_CRYPT_10700] Definition of API class ara::crypto::Serializable

Upstream requirements: RS_CRYPTO_02105

⌈

Kind:	class
Header file:	#include "ara/crypto/common/serializable.h"
Forwarding header file:	#include "ara/crypto/crypto_fwd.h"

▽

△

Scope:	namespace ara::crypto
Symbol:	Serializable
Syntax:	`class Serializable {...};`
Description:	Serializable object interface.

⌋

8.7.3.1 Public Member Functions

8.7.3.1.1 Special Member Functions

8.7.3.1.1.1 Move Constructor

[SWS_CRYPT_40998] Definition of API function ara::crypto::Serializable::Serializable

Upstream requirements: RS_CRYPTO_02004

⌈

Kind:	function
Header file:	#include "ara/crypto/common/serializable.h"
Scope:	`class ara::crypto::Serializable`
Syntax:	`Serializable (Serializable &&)=delete;`
Description:	Move-Constructor.

⌋

8.7.3.1.1.2 Copy Constructor

[SWS_CRYPT_40997] Definition of API function ara::crypto::Serializable::Serializable

Upstream requirements: RS_CRYPTO_02004

⌈

Kind:	function
Header file:	#include "ara/crypto/common/serializable.h"
Scope:	`class ara::crypto::Serializable`
Syntax:	`Serializable (const Serializable &)=delete;`
Description:	Copy-Constructor.

⌋

8.7.3.1.1.3 Copy Assignment Operator

[SWS_CRYPT_30204] Definition of API function ara::crypto::Serializable::operator=

Upstream requirements: RS_CRYPTO_02004

⌈

Kind:	function
Header file:	#include "ara/crypto/common/serializable.h"
Scope:	`class ara::crypto::Serializable`
Syntax:	`Serializable & operator= (const Serializable &other)=delete;`
Description:	Copy-assign another Serializable to this instance.

⌋

8.7.3.1.1.4 Move Assignment Operator

[SWS_CRYPT_30205] Definition of API function ara::crypto::Serializable::operator=

Upstream requirements: RS_CRYPTO_02004

⌈

Kind:	function
Header file:	#include "ara/crypto/common/serializable.h"
Scope:	`class ara::crypto::Serializable`
Syntax:	`Serializable & operator= (Serializable &&other)=delete;`
Description:	Move-assign another Serializable to this instance.

⌋

8.7.3.1.1.5 Destructor

[SWS_CRYPT_10710] Definition of API function ara::crypto::Serializable::~Serializable

Upstream requirements: RS_CRYPTO_02004, RS_CRYPTO_02302

⌈

Kind:	function
Header file:	#include "ara/crypto/common/serializable.h"
Scope:	`class ara::crypto::Serializable`
Syntax:	`virtual ~Serializable () noexcept=default;`
Exception Safety:	exception safe
Description:	Destructor.

⌋

8.7.3.1.2 Member Functions

8.7.3.1.2.1 ExportPublicly

[SWS_CRYPT_10711] Definition of API function ara::crypto::Serializable::ExportPublicly

Upstream requirements: RS_CRYPTO_02112

⌈

Kind:	function	
Header file:	#include "ara/crypto/common/serializable.h"	
Scope:	`class ara::crypto::Serializable`	
Syntax:	`virtual ara::core::Result< std::size_t > ExportPublicly (ReadWriteMemRegion out, FormatId format) const noexcept=0;`	
Parameters (in):	format	identifier of the output format
Parameters (out):	out	an output buffer that the serialized object will be written to
Return value:	ara::core::Result< std::size_t >	A result containing the size in bytes of the serialized object or an error.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kInsufficientCapacity	rollback_semantics
		if the output buffer is too small
	ara::crypto::CryptoErrc::kUnsupportedFormat	rollback_semantics
		if the specified format ID is not supported for this object type
	ara::crypto::CryptoErrc::kUnsupported	rollback_semantics
		if exporting the object is not supported
	ara::crypto::CryptoErrc::kUsageViolation	rollback_semantics
		if the object to be exported is a key and the allowed usage of the key does not include kAllowPlaintextExport
Description:	Serialize object using the provided format.	

⌋

8.7.3.1.2.2 GetExportSize

[SWS_CRYPT_50006] Definition of API function ara::crypto::Serializable::GetExportSize

Upstream requirements: RS_CRYPTO_02112

⌈

Kind:	function	
Header file:	#include "ara/crypto/common/serializable.h"	
Scope:	`class ara::crypto::Serializable`	
Syntax:	`virtual ara::core::Result< std::size_t > GetExportSize (FormatId format) const noexcept=0;`	
Parameters (in):	format	identifier of the output format

▽

△

Return value:	ara::core::Result< std::size_t >	A result containing the size in bytes of the serialized object or an error.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kUnsupportedFormat	rollback_semantics
		if the specified format ID is not supported for this object type
Description:	Get the size in bytes of the object when serialized in the provided format using ExportPublicly()	

⌋

8.8 Header: ara/crypto/cryp/auth_cipher_ctx.h

8.8.1 Class: AuthCipherCtx

[SWS_CRYPT_20100] Definition of API class ara::crypto::cryp::AuthCipherCtx

Upstream requirements: RS_CRYPTO_02207

⌈

Kind:	class
Header file:	#include "ara/crypto/cryp/auth_cipher_ctx.h"
Forwarding header file:	#include "ara/crypto/crypto_fwd.h"
Scope:	namespace ara::crypto::cryp
Symbol:	AuthCipherCtx
Base class:	CryptoContext
Syntax:	`class AuthCipherCtx : public CryptoContext {...};`
Description:	Authenticated Cipher Context interface.

⌋

8.8.1.1 Public Member Types

8.8.1.1.1 Type Alias: Sptr

[SWS_CRYPT_20101] Definition of API type ara::crypto::cryp::AuthCipherCtx::Sptr

Upstream requirements: RS_CRYPTO_02207

⌈

Kind:	type alias
Header file:	#include "ara/crypto/cryp/auth_cipher_ctx.h"
Scope:	`class ara::crypto::cryp::AuthCipherCtx`
Symbol:	Sptr

▽

△

Syntax:	`using Sptr = std::shared_ptr<AuthCipherCtx>;`
Description:	Smart pointer of the interface.

⌋

8.8.1.2 Public Member Functions

8.8.1.2.1 Member Functions

8.8.1.2.1.1 Cancel

[SWS_CRYPT_50077] Definition of API function ara::crypto::cryp::AuthCipherCtx::Cancel

Upstream requirements: RS_CRYPTO_02302

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/auth_cipher_ctx.h"	
Scope:	`class ara::crypto::cryp::AuthCipherCtx`	
Syntax:	`virtual ara::core::Result< void > Cancel () noexcept=0;`	
Return value:	ara::core::Result< void >	an empty result if the ongoing streaming operation was cancelled successfully, or a result containing an error
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kProcessingNotStarted	rollback_semantics
		if data processing was not started by a call of the `Start()` method
Description:	Cancel the ongoing streaming operation that has been initiated by Start() and has not yet been finished.	

⌋

8.8.1.2.1.2 EncryptSingleCallWithOutputIv

[SWS_CRYPT_50080] Definition of API function ara::crypto::cryp::AuthCipherCtx::EncryptSingleCallWithOutputIv

Upstream requirements: RS_CRYPTO_02302

⌈

Kind:	function
Header file:	#include "ara/crypto/cryp/auth_cipher_ctx.h"
Scope:	`class ara::crypto::cryp::AuthCipherCtx`

▽

△

Syntax:	`virtual ara::core::Result< std::size_t > EncryptSingleCallWithOutputIv (ReadOnlyMemRegion dataIn, ReadWriteMemRegion dataOut, ReadWriteMem Region tag, ReadWriteMemRegion ivOut, ara::core::Optional< ReadOnlyMem Region > authenticatedData) noexcept=0;`	
Parameters (in):	dataIn	an input data buffer containing the plaintext/ciphertext
	authenticatedData	optional additional authenticated data
Parameters (out):	dataOut	an output data buffer that contains en-/decrypted data
	ivOut	an output data buffer that the initialization vector (IV) generated by the context will be written to
Parameters (inout):	tag	for decryption: an input data buffer that contains the authentication tag. for encryption: an output data buffer that the authentication tag is written to.
Return value:	ara::core::Result< std::size_t >	a result containing the size of the output data or an error. The output data may be smaller than the size of the output buffer `out`.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kEmptyKeySlot	rollback_semantics
		if a KeySlot was configured that is currently is empty
	ara::crypto::CryptoErrc::kUnsupported	rollback_semantics
		if single shot processing is not supported by the context
	ara::crypto::CryptoErrc::kUnsupported	rollback_semantics
		if the context requires setting an IV
	ara::crypto::CryptoErrc::kUninitializedContext	rollback_semantics
		if `SetKey` has not been successfully called before
	ara::crypto::CryptoErrc::kBusyResource	rollback_semantics
		if a KeySlot has been configured, but is currently busy
	ara::crypto::CryptoErrc::kUsageViolation	rollback_semantics
		if the allowed usage of the SymmetricKey deployed or contained in the provided KeySlot at the time of invocation of this interface does not include kAllowDataEncrypt for encryption, or kAllowData Decrypt for decryption, respectively
	ara::crypto::CryptoErrc::kIncompatibleObject	rollback_semantics
		if a KeySlot has been configured, but does not contain a Symmetric Key
	ara::crypto::CryptoErrc::kIncompatibleObject	rollback_semantics
		if a KeySlot has been configured, but the SymmetricKey contained is not supported by this context
	ara::crypto::CryptoErrc::kInvalidInputSize	rollback_semantics
		if the size of provided input buffer or authentication tag is invalid
	ara::crypto::CryptoErrc::kInsufficientCapacity	rollback_semantics
		if the data or IV output buffer is too small
	ara::crypto::CryptoErrc::kAuthTagNotValid	rollback_semantics
		if the `transformDirection` is kDecrypt and the processed data cannot be authenticated
Description:	Perform encryption and calculate/verify of the authentication tag in a single-call and return an IV generated by the context.	

⌋

8.8.1.2.1.3 Finish

[SWS_CRYPT_50075] Definition of API function ara::crypto::cryp::AuthCipherCtx::Finish

Upstream requirements: RS_CRYPTO_02302

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/auth_cipher_ctx.h"	
Scope:	`class ara::crypto::cryp::AuthCipherCtx`	
Syntax:	`virtual ara::core::Result< std::size_t > Finish (ReadOnlyMemRegion in, ReadWriteMemRegion out, ReadWriteMemRegion tag) noexcept=0;`	
Parameters (in):	in	an input data buffer that contains a part of the plaintext/ciphertext
Parameters (out):	out	an output data buffer that the en-/decrypted data is written to
Parameters (inout):	tag	for decryption: an input data buffer that contains the authentication tag for encryption: an output data buffer that the authentication tag is written to
Return value:	ara::core::Result< std::size_t >	a result containing the size of the output data or an error. The output data may be smaller than the size of the output buffer `out`
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kProcessingNotStarted	rollback_semantics
		if data processing has not been started by a call of the `Start()` method
	ara::crypto::CryptoErrc::kInsufficientCapacity	rollback_semantics
		if the output buffer is too small
	ara::crypto::CryptoErrc::kInvalidInputSize	rollback_semantics
		if the size of the input buffer or the authentication tag is invalid
	ara::crypto::CryptoErrc::kAuthTagNotValid	rollback_semantics
		if the processed data cannot be authenticated during decryption
	ara::crypto::CryptoErrc::kUnsupported	rollback_semantics
		if stream processing is not supported by the context or
Description:	Finalize the en-/decryption and calculate/verify the authentication tag without returning an IV.	

⌋

8.8.1.2.1.4 FinishEncrypt

[SWS_CRYPT_50076] Definition of API function ara::crypto::cryp::AuthCipherCtx::FinishEncrypt

Upstream requirements: RS_CRYPTO_02302

⌈

Kind:	function
Header file:	#include "ara/crypto/cryp/auth_cipher_ctx.h"
Scope:	`class ara::crypto::cryp::AuthCipherCtx`

▽

△

Syntax:	`virtual ara::core::Result< std::size_t > FinishEncrypt (ReadOnlyMemRegion in, ReadWriteMemRegion out, ReadWriteMemRegion tag, ReadWriteMemRegion ivOut) noexcept=0;`	
Parameters (in):	in	an input data buffer that contains a part of the plaintext/ciphertext
Parameters (out):	out	an output data buffer that the en-/decrypted data is written to
	ivOut	an output data buffer that contains the IV
Parameters (inout):	tag	for decryption: an input data buffer that contains the authentication tag for encryption: an output data buffer that the authentication tag is written to
Return value:	ara::core::Result< std::size_t >	a result containing the size of the output data or an error. The output data may be smaller than the size of the output buffer `out`
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kProcessingNotStarted	rollback_semantics
		if data processing has not been started by a call of the `Start()` method
	ara::crypto::CryptoErrc::kInsufficientCapacity	rollback_semantics
		if the output buffer is too small
	ara::crypto::CryptoErrc::kInvalidInputSize	rollback_semantics
		if the size of the input buffer or the authentication tag is invalid
	ara::crypto::CryptoErrc::kAuthTagNotValid	rollback_semantics
		if the processed data cannot be authenticated during decryption
	ara::crypto::CryptoErrc::kUnsupported	rollback_semantics
		if stream processing is not supported by the context or if this context does not use an IV or if the IV has been provided by the application in the Start() function
Description:	Finalize the encryption and calculate the authentication tag and return the IV generated by the context.	

⌋

8.8.1.2.1.5 GetMaxAssociatedDataSize

[SWS_CRYPT_20103] Definition of API function ara::crypto::cryp::AuthCipherCtx::GetMaxAssociatedDataSize

Upstream requirements: RS_CRYPTO_02309

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/auth_cipher_ctx.h"	
Scope:	`class ara::crypto::cryp::AuthCipherCtx`	
Syntax:	`virtual ara::core::Result< std::size_t > GetMaxAssociatedDataSize () const noexcept=0;`	
Return value:	ara::core::Result< std::size_t >	maximal supported size of associated public data in bytes
Exception Safety:	exception safe	
Thread Safety:	thread-safe	

▽

△

Errors:	ara::crypto::CryptoErrc::kUnsupported	rollback_semantics
		if this interface is not supported
Description:	Get maximum supported size of associated public data.	

⌋

8.8.1.2.1.6 GetMaximumChunkSize

[SWS_CRYPT_50082] Definition of API function ara::crypto::cryp::AuthCipherCtx::GetMaximumChunkSize

Upstream requirements: RS_CRYPTO_02309

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/auth_cipher_ctx.h"	
Scope:	`class ara::crypto::cryp::AuthCipherCtx`	
Syntax:	`virtual ara::core::Result< std::size_t > GetMaximumChunkSize () const noexcept=0;`	
Return value:	ara::core::Result< std::size_t >	the maximum buffer size in bytes
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kUnsupported	rollback_semantics
		if this interface is not supported
Description:	Get the maximum input buffer size supported for processing in single calls to Update(), Finish(), FinishEncrypt() or ProcessSingleCall() by this context.	

⌋

8.8.1.2.1.7 GetMaximumInputIvSize

[SWS_CRYPT_50164] Definition of API function ara::crypto::cryp::AuthCipherCtx::GetMaximumInputIvSize

Upstream requirements: RS_CRYPTO_02309

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/auth_cipher_ctx.h"	
Scope:	`class ara::crypto::cryp::AuthCipherCtx`	
Syntax:	`virtual ara::core::Result< std::size_t > GetMaximumInputIvSize (CryptoTransform transformDirection) const noexcept=0;`	
Parameters (in):	transformDirection	the transformation direction, i.e., encryption or decryption
Return value:	ara::core::Result< std::size_t >	size of IV in bytes

▽

△

Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kUnsupported	rollback_semantics if this interface is not supported
Description:	Get the maximum IV size required to be passed to the context in the functions Start() and ProcessSingleCallWithInputIv() for the given transformDirection.	

⌋

8.8.1.2.1.8 GetMinimumChunkSize

[SWS_CRYPT_50162] Definition of API function ara::crypto::cryp::AuthCipherCtx::GetMinimumChunkSize

Upstream requirements: RS_CRYPTO_02309

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/auth_cipher_ctx.h"	
Scope:	`class ara::crypto::cryp::AuthCipherCtx`	
Syntax:	`virtual ara::core::Result< std::size_t > GetMinimumChunkSize () const noexcept=0;`	
Return value:	ara::core::Result< std::size_t >	the minimum buffer size in bytes
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kUnsupported	rollback_semantics if this interface is not supported
Description:	Get the minimum input buffer size supported for processing in single calls to Update(), Finish(), FinishEncrypt() or ProcessSingleCall() by this context.	

⌋

8.8.1.2.1.9 GetMinimumInputIvSize

[SWS_CRYPT_50163] Definition of API function ara::crypto::cryp::AuthCipherCtx::GetMinimumInputIvSize

Upstream requirements: RS_CRYPTO_02309

⌈

Kind:	function
Header file:	#include "ara/crypto/cryp/auth_cipher_ctx.h"
Scope:	`class ara::crypto::cryp::AuthCipherCtx`
Syntax:	`virtual ara::core::Result< std::size_t > GetMinimumInputIvSize (CryptoTransform transformDirection) const noexcept=0;`

▽

△

Parameters (in):	transformDirection	the transformation direction, i.e., encryption or decryption
Return value:	ara::core::Result< std::size_t >	size of IV in bytes
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kUnsupported	rollback_semantics
		if this interface is not supported
Description:	Get the minimum IV size required to be passed to the context in the functions Start() and ProcessSingleCallWithInputIv() for the given transformDirection.	

⌋

8.8.1.2.1.10 GetOutputIvSize

[SWS_CRYPT_50165] Definition of API function ara::crypto::cryp::AuthCipherCtx::GetOutputIvSize

Upstream requirements: RS_CRYPTO_02309

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/auth_cipher_ctx.h"	
Scope:	`class ara::crypto::cryp::AuthCipherCtx`	
Syntax:	`virtual ara::core::Result< std::size_t > GetOutputIvSize () const noexcept=0;`	
Return value:	ara::core::Result< std::size_t >	size of IV in bytes
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kUnsupported	rollback_semantics
		if this interface is not supported
Description:	Get the output IV size that will be returned by the functions FinishEncrypt() (in case no IV has been provided by the application during the call to Start()), and EncryptSingleCallWithOutputIv().	

⌋

8.8.1.2.1.11 GetRequiredOutputBufferSize

[SWS_CRYPT_50081] Definition of API function ara::crypto::cryp::AuthCipherCtx::GetRequiredOutputBufferSize

Upstream requirements: RS_CRYPTO_02302

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/auth_cipher_ctx.h"	
Scope:	`class ara::crypto::cryp::AuthCipherCtx`	
Syntax:	`virtual ara::core::Result< std::size_t > GetRequiredOutputBufferSize (std::size_t inputSize, bool isFinal) const noexcept;`	
Parameters (in):	inputSize	size of input data
	isFinal	if `true`, calculate maximum required output buffer size for Finish(), FinishEncrypt() and ProcessSingleCall() otherwise for Update()
Return value:	ara::core::Result< std::size_t >	a result containing the required capacity of the output buffer (in bytes) or an error
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kUninitializedContext	rollback_semantics
		if the context was not initialized by deploying a key and setting the transformation direction
	ara::crypto::CryptoErrc::kUnsupported	rollback_semantics
		if this interface is not supported
Description:	Calculate the maximum required size of the output buffer depending on the input size in the current state of the context.	

⌋

8.8.1.2.1.12 GetTagSize

[SWS_CRYPT_50161] Definition of API function ara::crypto::cryp::AuthCipherCtx::GetTagSize

Upstream requirements: RS_CRYPTO_02309

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/auth_cipher_ctx.h"	
Scope:	`class ara::crypto::cryp::AuthCipherCtx`	
Syntax:	`virtual ara::core::Result< std::size_t > GetTagSize () const noexcept=0;`	
Return value:	ara::core::Result< std::size_t >	size of the authentication tag in bytes
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kUnsupported	rollback_semantics
		if this interface is not supported

▽

△

Description:	Get the output tag size.

⌋

8.8.1.2.1.13 ProcessSingleCall

[SWS_CRYPT_50078] Definition of API function ara::crypto::cryp::AuthCipherCtx::ProcessSingleCall

Upstream requirements: RS_CRYPTO_02302

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/auth_cipher_ctx.h"	
Scope:	`class ara::crypto::cryp::AuthCipherCtx`	
Syntax:	`virtual ara::core::Result< std::size_t > ProcessSingleCall (ReadOnlyMemRegion dataIn, ReadWriteMemRegion dataOut, ReadWriteMemRegion tag, CryptoTransform transformDirection, ara::core::Optional< ReadOnlyMemRegion > authenticatedData) noexcept=0;`	
Parameters (in):	dataIn	an input data buffer containing the plaintext/ciphertext
	transformDirection	the transformation direction, i.e., encryption or decryption
	authenticatedData	optional additional authenticated data
Parameters (out):	dataOut	an output data buffer that contains en-/decrypted data
Parameters (inout):	tag	for decryption: an input data buffer that contains the authentication tag. for encryption: an output data buffer that the authentication tag is written to.
Return value:	ara::core::Result< std::size_t >	a result containing the size of the output data or an error. The output data may be smaller than the size of the output buffer `out`.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kEmptyKeySlot	rollback_semantics
		if a KeySlot was configured that is currently is empty
	ara::crypto::CryptoErrc::kUnsupported	rollback_semantics
		if single shot processing is not supported by the context
	ara::crypto::CryptoErrc::kUnsupported	rollback_semantics
		if the context requires setting or returning an IV
	ara::crypto::CryptoErrc::kUninitializedContext	rollback_semantics
		if `SetKey` has not been successfully called before
	ara::crypto::CryptoErrc::kBusyResource	rollback_semantics
		if a KeySlot has been configured, but is currently busy
	ara::crypto::CryptoErrc::kUsageViolation	rollback_semantics
		if the allowed usage of the SymmetricKey deployed or contained in the provided KeySlot at the time of invocation of this interface does not include kAllowDataEncrypt for encryption, or kAllowDataDecrypt for decryption, respectively
	ara::crypto::CryptoErrc::kIncompatibleObject	rollback_semantics

▽

△

		if a KeySlot has been configured, but does not contain a Symmetric Key
	ara::crypto::CryptoErrc::kIncompatibleObject	rollback_semantics
		if a KeySlot has been configured, but the SymmetricKey contained is not supported by this context
	ara::crypto::CryptoErrc::kInvalidInputSize	rollback_semantics
		if the size of provided input buffer or authentication tag is invalid
	ara::crypto::CryptoErrc::kInsufficientCapacity	rollback_semantics
		if the output buffer is too small
	ara::crypto::CryptoErrc::kAuthTagNotValid	rollback_semantics
		if the `transformDirection` is kDecrypt and the processed data cannot be authenticated
Description:	Perform en-/decryption and calculate/verify of the authentication tag in a single-call without setting or returning an IV.	

⌋

8.8.1.2.1.14 ProcessSingleCallWithInputIv

[SWS_CRYPT_50079] Definition of API function ara::crypto::cryp::AuthCipherCtx::ProcessSingleCallWithInputIv

Upstream requirements: RS_CRYPTO_02302

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/auth_cipher_ctx.h"	
Scope:	`class ara::crypto::cryp::AuthCipherCtx`	
Syntax:	`virtual ara::core::Result< std::size_t > ProcessSingleCallWithInputIv (ReadOnlyMemRegion dataIn, ReadWriteMemRegion dataOut, ReadWriteMemRegion tag, CryptoTransform transformDirection, ReadOnlyMemRegion iv, ara::core::Optional< ReadOnlyMemRegion > authenticatedData) noexcept=0;`	
Parameters (in):	dataIn	an input data buffer containing the plaintext/ciphertext
	transformDirection	the transformation direction, i.e., encryption or decryption
	iv	an Initialization Vector (IV) or "nonce" value
	authenticatedData	optional additional authenticated data
Parameters (out):	dataOut	an output data buffer that contains en-/decrypted data
Parameters (inout):	tag	for decryption: an input data buffer that contains the authentication tag. for encryption: an output data buffer that the authentication tag is written to.
Return value:	ara::core::Result< std::size_t >	a result containing the size of the output data or an error. The output data may be smaller than the size of the output buffer `out`.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kEmptyKeySlot	rollback_semantics
		if a KeySlot was configured that is currently is empty

▽

△

	ara::crypto::CryptoErrc::kUnsupported	rollback_semantics
		if single shot processing is not supported by the context
	ara::crypto::CryptoErrc::kUnsupported	rollback_semantics
		if the context requires setting or returning an IV
	ara::crypto::CryptoErrc::kUninitializedContext	rollback_semantics
		if `SetKey` has not been successfully called before
	ara::crypto::CryptoErrc::kBusyResource	rollback_semantics
		if a KeySlot has been configured, but is currently busy
	ara::crypto::CryptoErrc::kUsageViolation	rollback_semantics
		if the allowed usage of the SymmetricKey deployed or contained in the provided KeySlot at the time of invocation of this interface does not include kAllowDataEncrypt for encryption, or kAllowData Decrypt for decryption, respectively
	ara::crypto::CryptoErrc::kIncompatibleObject	rollback_semantics
		if a KeySlot has been configured, but does not contain a Symmetric Key
	ara::crypto::CryptoErrc::kIncompatibleObject	rollback_semantics
		if a KeySlot has been configured, but the SymmetricKey contained is not supported by this context
	ara::crypto::CryptoErrc::kInvalidInputSize	rollback_semantics
		if the size of provided input buffer, authentication tag, or IV is invalid
	ara::crypto::CryptoErrc::kInsufficientCapacity	rollback_semantics
		if the data or IV output buffer is too small
	ara::crypto::CryptoErrc::kAuthTagNotValid	rollback_semantics
		if the `transformDirection` is kDecrypt and the processed data cannot be authenticated
Description:	Perform en-/decryption and calculate/verify of the authentication tag in a single-call, providing an IV to be used during en-/decryption.	

⌋

8.8.1.2.1.15 SetKey

[SWS_CRYPT_50072] Definition of API function ara::crypto::cryp::AuthCipherCtx::SetKey

Upstream requirements: RS_CRYPTO_02001, RS_CRYPTO_02003

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/auth_cipher_ctx.h"	
Scope:	`class ara::crypto::cryp::AuthCipherCtx`	
Syntax:	`virtual ara::core::Result< void > SetKey (KeySlot::Sptr keySlot) noexcept=0;`	
Parameters (in):	keySlot	a keyslot containing a symmetric key
Return value:	ara::core::Result< void >	an empty result or an error
Exception Safety:	exception safe	

▽

△

Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kUnsupported	rollback_semantics
		if the context only supports setting a SymmetricKey
Description:	The interface shall configure the context to use the key stored in the provided KeySlot when the authenticated encrypt or decrypt operation is executed.	

⌋

8.8.1.2.1.16 SetKey

[SWS_CRYPT_23911] Definition of API function ara::crypto::cryp::AuthCipherCtx::SetKey

Upstream requirements: RS_CRYPTO_02001, RS_CRYPTO_02003

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/auth_cipher_ctx.h"	
Scope:	`class ara::crypto::cryp::AuthCipherCtx`	
Syntax:	`virtual ara::core::Result< void > SetKey (SymmetricKey::Sptr key) noexcept=0;`	
Parameters (in):	key	symmetric key object used for authenticated en-/decryption
Return value:	ara::core::Result< void >	an empty result or an error
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kUnsupported	rollback_semantics
		if the context only supports setting a KeySlot
	ara::crypto::CryptoErrc::kIncompatibleObject	rollback_semantics
		if the provided SymmetricKey is incompatible with this AuthCipherCtx
	ara::crypto::CryptoErrc::kUsageViolation	rollback_semantics
		if the allowed usage of the provided SymmetricKey does not include kAllowDataEncrypt for encryption, or kAllowDataDecrypt for decryption, respectively
Description:	The interface shall configure the context to use the provided SymmetricKey.	

⌋

8.8.1.2.1.17 Start

[SWS_CRYPT_24715] Definition of API function ara::crypto::cryp::AuthCipherCtx::Start

Upstream requirements: RS_CRYPTO_02302

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/auth_cipher_ctx.h"	
Scope:	`class ara::crypto::cryp::AuthCipherCtx`	
Syntax:	`virtual ara::core::Result< void > Start (CryptoTransform transform Direction, ReadOnlyMemRegion iv) noexcept=0;`	
Parameters (in):	transformDirection	the transformation direction, i.e., encryption or decryption
	iv	an Initialization Vector (IV) or "nonce" value. The allowed size of the IV can be queried using the functions GetMinimumInputIvSize() and GetMaximumInputIvSize().
Return value:	ara::core::Result< void >	an empty result or an error
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kEmptyKeySlot	rollback_semantics
		if a KeySlot was configured that is currently is empty
	ara::crypto::CryptoErrc::kUnsupported	rollback_semantics
		if stream processing is not supported by the context
	ara::crypto::CryptoErrc::kUnsupported	rollback_semantics
		if the context does not support setting an IV in the given transform Direction
	ara::crypto::CryptoErrc::kUninitializedContext	rollback_semantics
		if `SetKey` has not been successfully called before
	ara::crypto::CryptoErrc::kBusyResource	rollback_semantics
		if the configured KeySlot is currently busy
	ara::crypto::CryptoErrc::kUsageViolation	rollback_semantics
		if the allowed usage of the SymmetricKey deployed or contained in the provided KeySlot at the time of invocation of this interface does not include kAllowDataEncrypt for encryption, or kAllowData Decrypt for decryption, respectively
	ara::crypto::CryptoErrc::kIncompatibleObject	rollback_semantics
		if a KeySlot has been configured, but does not contain a Symmetric Key
	ara::crypto::CryptoErrc::kIncompatibleObject	rollback_semantics
		if a KeySlot has been configured, but the SymmetricKey contained is not supported by this context
	ara::crypto::CryptoErrc::kInvalidInputSize	rollback_semantics
		if the size of provided IV is not supported
Description:	Initialize the context for a new encryption or decryption and set the IV used by the context.	

⌋

8.8.1.2.1.18 Start

[SWS_CRYPT_50073] Definition of API function ara::crypto::cryp::AuthCipherCtx::Start

Upstream requirements: RS_CRYPTO_02302

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/auth_cipher_ctx.h"	
Scope:	`class ara::crypto::cryp::AuthCipherCtx`	
Syntax:	`virtual ara::core::Result< void > Start (CryptoTransform transform Direction) noexcept=0;`	
Parameters (in):	transformDirection	the transformation direction, i.e., encryption or decryption
Return value:	ara::core::Result< void >	an empty result or an error
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kEmptyKeySlot	rollback_semantics
		if a KeySlot was configured that is currently is empty
	ara::crypto::CryptoErrc::kUnsupported	rollback_semantics
		if stream processing is not supported by the context
	ara::crypto::CryptoErrc::kUnsupported	rollback_semantics
		if the context requires setting an IV in the given transformDirection
	ara::crypto::CryptoErrc::kUninitializedContext	rollback_semantics
		if `SetKey` has not been successfully called before
	ara::crypto::CryptoErrc::kBusyResource	rollback_semantics
		if a KeySlot has been configured, but is currently busy
	ara::crypto::CryptoErrc::kUsageViolation	rollback_semantics
		if the allowed usage of the SymmetricKey deployed or contained in the provided KeySlot at the time of invocation of this interface does not include kAllowDataEncrypt for encryption, or kAllowData Decrypt for decryption, respectively
	ara::crypto::CryptoErrc::kIncompatibleObject	rollback_semantics
		if a KeySlot has been configured, but does not contain a Symmetric Key
	ara::crypto::CryptoErrc::kIncompatibleObject	rollback_semantics
		if a KeySlot has been configured, but the SymmetricKey contained is not supported by this context
Description:	Initialize the context for a new encryption or decryption. An IV is either not required for this context in this transformation direction, or set by the context itself.	

⌋

8.8.1.2.1.19 UpdateAuthenticatedData

[SWS_CRYPT_20313] Definition of API function ara::crypto::cryp::AuthCipherCtx::UpdateAuthenticatedData

Upstream requirements: RS_CRYPTO_02302

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/auth_cipher_ctx.h"	
Scope:	`class ara::crypto::cryp::AuthCipherCtx`	
Syntax:	`virtual ara::core::Result< void > UpdateAuthenticatedData (ReadOnlyMemRegion authenticatedData) noexcept=0;`	
Parameters (in):	authenticatedData	a part of the additional authenticated data
Return value:	ara::core::Result< void >	an empty result or an error
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kProcessingNotStarted	rollback_semantics
		if the processing was not initiated by a call of the `Start()` method
	ara::crypto::CryptoErrc::kInvalidUsageOrder	rollback_semantics
		if UpdateConfidentialData has already been called
	ara::crypto::CryptoErrc::kInvalidInputSize	rollback_semantics
		if the size of the input buffer is invalid
	ara::crypto::CryptoErrc::kUnsupported	rollback_semantics
		if stream processing is not supported by the context
Description:	Update the tag calculation by a new chunk of additional authenticated data.	

⌋

8.8.1.2.1.20 UpdateConfidentialData

[SWS_CRYPT_50074] Definition of API function ara::crypto::cryp::AuthCipherCtx::UpdateConfidentialData

Upstream requirements: RS_CRYPTO_02302

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/auth_cipher_ctx.h"	
Scope:	`class ara::crypto::cryp::AuthCipherCtx`	
Syntax:	`virtual ara::core::Result< std::size_t > UpdateConfidentialData (ReadOnlyMemRegion in, ReadWriteMemRegion out) noexcept=0;`	
Parameters (in):	in	an input data buffer containing a part of the ciphertext or plaintext
Parameters (out):	out	an output data buffer that the encrypted or decrypted data is written too
Return value:	ara::core::Result< std::size_t >	a result containing the size of the output data or an error. The output data may be smaller than the size of the output buffer `out`.
Exception Safety:	exception safe	

▽

△

Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kProcessingNotStarted	rollback_semantics
		if data processing was not started by a call of the `Start()` method
	ara::crypto::CryptoErrc::kInsufficientCapacity	rollback_semantics
		if the output buffer is too small
	ara::crypto::CryptoErrc::kInvalidInputSize	rollback_semantics
		if the size of the input buffer is invalid
	ara::crypto::CryptoErrc::kUnsupported	rollback_semantics
		if stream processing is not supported by the context
Description:	Encrypt or decrypt confidential data. When used for decryption, the plaintext is returned without verifying its authenticity. An application must not use the data before verifying its authenticity using Finish().	

⌋

8.9 Header: ara/crypto/cryp/crypto_context.h

8.9.1 Class: CryptoContext

[SWS_CRYPT_20400] Definition of API class ara::crypto::cryp::CryptoContext

Upstream requirements: RS_CRYPTO_02008

⌈

Kind:	class
Header file:	#include "ara/crypto/cryp/crypto_context.h"
Forwarding header file:	#include "ara/crypto/crypto_fwd.h"
Scope:	namespace ara::crypto::cryp
Symbol:	CryptoContext
Syntax:	`class CryptoContext {...};`
Description:	A common interface of a mutable cryptographic context, i.e. that is not binded to a single crypto object.

⌋

8.9.1.1 Public Member Functions

8.9.1.1.1 Special Member Functions

8.9.1.1.1.1 Copy Constructor

[SWS_CRYPT_41003] Definition of API function ara::crypto::cryp::CryptoContext::CryptoContext

Upstream requirements: RS_CRYPTO_02004

⌈

Kind:	function
Header file:	#include "ara/crypto/cryp/crypto_context.h"
Scope:	`class ara::crypto::cryp::CryptoContext`
Syntax:	`CryptoContext (const CryptoContext &)=delete;`
Description:	Copy-Constructor.

⌋

8.9.1.1.1.2 Move Constructor

[SWS_CRYPT_41004] Definition of API function ara::crypto::cryp::CryptoContext::CryptoContext

Upstream requirements: RS_CRYPTO_02004

⌈

Kind:	function
Header file:	#include "ara/crypto/cryp/crypto_context.h"
Scope:	`class ara::crypto::cryp::CryptoContext`
Syntax:	`CryptoContext (CryptoContext &&)=delete;`
Description:	Move-Constructor.

⌋

8.9.1.1.1.3 Copy Assignment Operator

[SWS_CRYPT_30214] Definition of API function ara::crypto::cryp::CryptoContext::operator=

Upstream requirements: RS_CRYPTO_02004

⌈

Kind:	function
Header file:	#include "ara/crypto/cryp/crypto_context.h"
Scope:	`class ara::crypto::cryp::CryptoContext`
Syntax:	`CryptoContext & operator= (const CryptoContext &other)=delete;`
Description:	Copy-assign another CryptoContext to this instance.

⌋

8.9.1.1.1.4 Move Assignment Operator

[SWS_CRYPT_30215] Definition of API function ara::crypto::cryp::CryptoContext::operator=

Upstream requirements: RS_CRYPTO_02004

⌈

Kind:	function
Header file:	#include "ara/crypto/cryp/crypto_context.h"
Scope:	`class ara::crypto::cryp::CryptoContext`
Syntax:	`CryptoContext & operator= (CryptoContext &&other)=delete;`
Description:	Move-assign another CryptoContext to this instance.

⌋

8.9.1.1.1.5 Destructor

[SWS_CRYPT_20401] Definition of API function ara::crypto::cryp::CryptoContext::~CryptoContext

Upstream requirements: RS_CRYPTO_02008

⌈

Kind:	function
Header file:	#include "ara/crypto/cryp/crypto_context.h"
Scope:	`class ara::crypto::cryp::CryptoContext`
Syntax:	`virtual ~CryptoContext () noexcept=default;`
Exception Safety:	exception safe
Description:	Destructor.

⌋

8.9.1.1.2 Member Functions

8.9.1.1.2.1 GetAlgId

[SWS_CRYPT_20411] Definition of API function ara::crypto::cryp::CryptoContext::GetAlgId

Upstream requirements: RS_CRYPTO_02008

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/crypto_context.h"	
Scope:	`class ara::crypto::cryp::CryptoContext`	
Syntax:	`virtual ara::core::Result< ara::crypto::AlgIdString > GetAlgId () const noexcept=0;`	
Return value:	ara::core::Result< ara::crypto::AlgIdString >	the AlgId of the context or an error
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	This function does not specify any standardized errors.	
Description:	The interface shall return the AlgId of the context.	

⌋

8.10 Header: ara/crypto/cryp/crypto_provider.h

8.10.1 Class: CryptoProvider

[SWS_CRYPT_20700] Definition of API class ara::crypto::cryp::CryptoProvider

Upstream requirements: RS_CRYPTO_02305, RS_CRYPTO_02307, RS_CRYPTO_02401

⌈

Kind:	class
Port Interfaces:	`CryptoProviderInterface`
Header file:	#include "ara/crypto/cryp/crypto_provider.h"
Forwarding header file:	#include "ara/crypto/crypto_fwd.h"
Scope:	namespace ara::crypto::cryp
Symbol:	CryptoProvider
Syntax:	`class CryptoProvider {...};`
Description:	`CryptoProviderInterface` representing one implementation of a set of cryptographic primitives with a key storage.

⌋

8.10.1.1 Public Member Types

8.10.1.1.1 Type Alias: Sptr

[SWS_CRYPT_20701] Definition of API type ara::crypto::cryp::CryptoProvider::Sptr

Upstream requirements: RS_CRYPTO_02109

⌈

Kind:	type alias
Header file:	#include "ara/crypto/cryp/crypto_provider.h"
Scope:	`class ara::crypto::cryp::CryptoProvider`
Symbol:	Sptr
Syntax:	`using Sptr = std::shared_ptr<CryptoProvider>;`
Description:	Shared pointer of the CryptoProvider.

⌋

8.10.1.2 Public Member Functions

8.10.1.2.1 Special Member Functions

8.10.1.2.1.1 Move Constructor

[SWS_CRYPT_41006] Definition of API function ara::crypto::cryp::CryptoProvider::CryptoProvider

Upstream requirements: RS_CRYPTO_02004

⌈

Kind:	function
Header file:	#include "ara/crypto/cryp/crypto_provider.h"
Scope:	`class ara::crypto::cryp::CryptoProvider`
Syntax:	`CryptoProvider (CryptoProvider &&)=delete;`
Description:	Move-Constructor.

⌋

8.10.1.2.1.2 Copy Constructor

[SWS_CRYPT_41005] Definition of API function ara::crypto::cryp::CryptoProvider::CryptoProvider

Upstream requirements: RS_CRYPTO_02004

⌈

Kind:	function
Header file:	#include "ara/crypto/cryp/crypto_provider.h"
Scope:	`class ara::crypto::cryp::CryptoProvider`
Syntax:	`CryptoProvider (const CryptoProvider &)=delete;`
Description:	Copy-Constructor.

⌋

8.10.1.2.1.3 Move Assignment Operator

[SWS_CRYPT_30217] Definition of API function ara::crypto::cryp::CryptoProvider::operator=

Upstream requirements: RS_CRYPTO_02004

⌈

Kind:	function
Header file:	#include "ara/crypto/cryp/crypto_provider.h"
Scope:	`class ara::crypto::cryp::CryptoProvider`
Syntax:	`CryptoProvider & operator= (CryptoProvider &&other)=delete;`
Description:	Move-assign another CryptoProvider to this instance.

⌋

8.10.1.2.1.4 Copy Assignment Operator

[SWS_CRYPT_30216] Definition of API function ara::crypto::cryp::CryptoProvider::operator=

Upstream requirements: RS_CRYPTO_02004

⌈

Kind:	function
Header file:	#include "ara/crypto/cryp/crypto_provider.h"
Scope:	`class ara::crypto::cryp::CryptoProvider`
Syntax:	`CryptoProvider & operator= (const CryptoProvider &other)=delete;`
Description:	Copy-assign another CryptoProvider to this instance.

⌋

8.10.1.2.1.5 Destructor

[SWS_CRYPT_20710] Definition of API function ara::crypto::cryp::CryptoProvider::~CryptoProvider

Upstream requirements: RS_CRYPTO_02107

⌈

Kind:	function
Header file:	#include "ara/crypto/cryp/crypto_provider.h"
Scope:	`class ara::crypto::cryp::CryptoProvider`
Syntax:	`virtual ~CryptoProvider () noexcept=default;`
Exception Safety:	exception safe
Description:	Destructor.

⌋

8.10.1.2.2 Member Functions

8.10.1.2.2.1 CreateAuthCipherCtx

[SWS_CRYPT_20745] Definition of API function ara::crypto::cryp::CryptoProvider::CreateAuthCipherCtx

Upstream requirements: RS_CRYPTO_02207, RS_AP_00144

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/crypto_provider.h"	
Scope:	`class ara::crypto::cryp::CryptoProvider`	
Syntax:	`virtual ara::core::Result< AuthCipherCtx::Sptr > CreateAuthCipherCtx (AlgId algId) noexcept=0;`	
Parameters (in):	algId	identifier of the target crypto algorithm
Return value:	ara::core::Result< AuthCipherCtx::Sptr >	Smart pointer to the created context
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kInvalidArgument	rollback_semantics
		if `algId` argument specifies a crypto algorithm different from symmetric authenticated stream cipher
	ara::crypto::CryptoErrc::kUnknownIdentifier	rollback_semantics
		if `algId` argument has an unsupported value
Description:	Create a symmetric authenticated cipher context.	

⌋

8.10.1.2.2.2 CreateDecryptorPrivateCtx

[SWS_CRYPT_20751] Definition of API function ara::crypto::cryp::CryptoProvider::CreateDecryptorPrivateCtx

Upstream requirements: RS_CRYPTO_02202, RS_AP_00144

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/crypto_provider.h"	
Scope:	`class ara::crypto::cryp::CryptoProvider`	
Syntax:	`virtual ara::core::Result< DecryptorPrivateCtx::Sptr > CreateDecryptorPrivateCtx (AlgId algId) noexcept=0;`	
Parameters (in):	algId	identifier of the target asymmetric encryption/decryption algorithm
Return value:	ara::core::Result< DecryptorPrivateCtx::Sptr >	Smart pointer to the created context
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kUnknownIdentifier	rollback_semantics
		if `AlgId` argument has an unsupported value
Description:	Create a decryption private key context.	

⌋

8.10.1.2.2.3 CreateEncryptorPublicCtx

[SWS_CRYPT_20750] Definition of API function ara::crypto::cryp::CryptoProvider::CreateEncryptorPublicCtx

Upstream requirements: RS_CRYPTO_02202, RS_AP_00144

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/crypto_provider.h"	
Scope:	`class ara::crypto::cryp::CryptoProvider`	
Syntax:	`virtual ara::core::Result< EncryptorPublicCtx::Sptr > CreateEncryptorPublicCtx (AlgId algId) noexcept=0;`	
Parameters (in):	algId	identifier of the target asymmetric encryption/decryption algorithm
Return value:	ara::core::Result< EncryptorPublicCtx::Sptr >	Smart pointer to the created context
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kUnknownIdentifier	rollback_semantics
		if `AlgId` argument has an unsupported value
Description:	Create an encryption public key context.	

⌋

8.10.1.2.2.4 CreateHashFunctionCtx

[SWS_CRYPT_20747] Definition of API function ara::crypto::cryp::CryptoProvider::CreateHashFunctionCtx

Upstream requirements: RS_CRYPTO_02205, RS_AP_00144

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/crypto_provider.h"	
Scope:	`class ara::crypto::cryp::CryptoProvider`	
Syntax:	`virtual ara::core::Result< HashFunctionCtx::Sptr > CreateHashFunctionCtx (AlgId algId) noexcept=0;`	
Parameters (in):	algId	identifier of the target crypto algorithm
Return value:	ara::core::Result< HashFunctionCtx::Sptr >	Smart pointer to the created context
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kInvalidArgument	rollback_semantics
		if `algId` argument specifies a crypto algorithm different from hash function
	ara::crypto::CryptoErrc::kUnknownIdentifier	rollback_semantics
		if `algId` argument has an unsupported value
Description:	Create a hash function context.	

⌋

8.10.1.2.2.5 CreateKeyAgreementPrivateCtx

[SWS_CRYPT_20758] Definition of API function ara::crypto::cryp::CryptoProvider::CreateKeyAgreementPrivateCtx

Upstream requirements: RS_CRYPTO_02104, RS_AP_00144

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/crypto_provider.h"	
Scope:	`class ara::crypto::cryp::CryptoProvider`	
Syntax:	`virtual ara::core::Result< KeyAgreementPrivateCtx::Sptr > CreateKeyAgreementPrivateCtx (AlgId algId) noexcept=0;`	
Parameters (in):	algId	identifier of the target key-agreement crypto algorithm
Return value:	ara::core::Result< KeyAgreementPrivateCtx::Sptr >	Smart pointer to the created context
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kInvalidArgument	rollback_semantics

▽

△

		if `algId` argument specifies a crypto algorithm different from key-agreement
	ara::crypto::CryptoErrc::kUnknownIdentifier	rollback_semantics
		if `algId` argument has an unsupported value
Description:	Create a key-agreement private key context.	

⌋

8.10.1.2.2.6 CreateKeyDecapsulatorPrivateCtx

[SWS_CRYPT_20753] Definition of API function ara::crypto::cryp::CryptoProvider::CreateKeyDecapsulatorPrivateCtx

Upstream requirements: RS_CRYPTO_02104, RS_CRYPTO_02209, RS_AP_00144

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/crypto_provider.h"	
Scope:	`class ara::crypto::cryp::CryptoProvider`	
Syntax:	`virtual ara::core::Result< KeyDecapsulatorPrivateCtx::Sptr > CreateKeyDecapsulatorPrivateCtx (AlgId algId) noexcept=0;`	
Parameters (in):	algId	identifier of the target KEM crypto algorithm
Return value:	ara::core::Result< KeyDecapsulatorPrivateCtx::Sptr >	Smart pointer to the created context
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kInvalidArgument	rollback_semantics
		if `algId` argument specifies a crypto algorithm different from asymmetric KEM
	ara::crypto::CryptoErrc::kUnknownIdentifier	rollback_semantics
		if `algId` argument has an unsupported value
Description:	Create a key-decapsulator private context of a Key Encapsulation Mechanism (KEM).	

⌋

8.10.1.2.2.7 CreateKeyDerivationFunctionCtx

[SWS_CRYPT_20748] Definition of API function ara::crypto::cryp::CryptoProvider::CreateKeyDerivationFunctionCtx

Upstream requirements: RS_CRYPTO_02103, RS_AP_00144

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/crypto_provider.h"	
Scope:	`class ara::crypto::cryp::CryptoProvider`	
Syntax:	`virtual ara::core::Result< KeyDerivationFunctionCtx::Sptr > CreateKeyDerivationFunctionCtx (AlgId algId) noexcept=0;`	
Parameters (in):	algId	identifier of the target crypto algorithm
Return value:	ara::core::Result< KeyDerivationFunctionCtx::Sptr >	Smart pointer to the created context
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kInvalidArgument	rollback_semantics
		if `algId` argument specifies a crypto algorithm different from key derivation function
	ara::crypto::CryptoErrc::kUnknownIdentifier	rollback_semantics
		if `algId` argument has an unsupported value
Description:	Create a key derivation function context.	

⌋

8.10.1.2.2.8 CreateKeyEncapsulatorPublicCtx

[SWS_CRYPT_20752] Definition of API function ara::crypto::cryp::CryptoProvider::CreateKeyEncapsulatorPublicCtx

Upstream requirements: RS_CRYPTO_02104, RS_CRYPTO_02209, RS_AP_00144

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/crypto_provider.h"	
Scope:	`class ara::crypto::cryp::CryptoProvider`	
Syntax:	`virtual ara::core::Result< KeyEncapsulatorPublicCtx::Sptr > CreateKeyEncapsulatorPublicCtx (AlgId algId) noexcept=0;`	
Parameters (in):	algId	identifier of the target KEM crypto algorithm
Return value:	ara::core::Result< KeyEncapsulatorPublicCtx::Sptr >	Smart pointer to the created context
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kInvalidArgument	rollback_semantics

▽

△

		if `algId` argument specifies a crypto algorithm different from asymmetric KEM
	ara::crypto::CryptoErrc::kUnknownIdentifier	rollback_semantics
		if `algId` argument has an unsupported value
Description:	Create a key-encapsulator public context of a Key Encapsulation Mechanism (KEM).	

⌋

8.10.1.2.2.9 CreateMessageAuthnCodeCtx

[SWS_CRYPT_20746] Definition of API function ara::crypto::cryp::CryptoProvider::CreateMessageAuthnCodeCtx

Upstream requirements: RS_CRYPTO_02203, RS_AP_00144

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/crypto_provider.h"	
Scope:	`class ara::crypto::cryp::CryptoProvider`	
Syntax:	`virtual ara::core::Result< MessageAuthnCodeCtx::Sptr > CreateMessageAuthnCodeCtx (AlgId algId) noexcept=0;`	
Parameters (in):	algId	identifier of the target crypto algorithm
Return value:	ara::core::Result< MessageAuthnCodeCtx::Sptr >	Smart pointer to the created context
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kInvalidArgument	rollback_semantics
		if `algId` argument specifies a crypto algorithm different from symmetric message authentication code
	ara::crypto::CryptoErrc::kUnknownIdentifier	rollback_semantics
		if `algId` argument has an unsupported value
Description:	Create a symmetric message authentication code context.	

⌋

8.10.1.2.2.10 CreateProtectedKeyImportCtx

[SWS_CRYPT_50017] Definition of API function ara::crypto::cryp::CryptoProvider::CreateProtectedKeyImportCtx

Upstream requirements: RS_CRYPTO_02105

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/crypto_provider.h"	
Scope:	`class ara::crypto::cryp::CryptoProvider`	
Syntax:	`virtual ara::core::Result< ProtectedKeyImportCtx::Sptr > CreateProtectedKeyImportCtx (AlgId algId) noexcept=0;`	
Parameters (in):	algId	identifier of the wrapping algorithm
Return value:	ara::core::Result< ProtectedKeyImport Ctx::Sptr >	Smart pointer to the created context
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::k InvalidArgument	rollback_semantics
		if `algId` argument specifies a crypto algorithm that is not supported by this CryptoProvider for wrapping
	ara::crypto::CryptoErrc::k UnknownIdentifier	rollback_semantics
		if `algId` argument has an unsupported value
Description:	The interface shall create a protected key import context.	

⌋

8.10.1.2.2.11 CreateRandomGeneratorCtx

[SWS_CRYPT_20741] Definition of API function ara::crypto::cryp::CryptoProvider::CreateRandomGeneratorCtx

Upstream requirements: RS_CRYPTO_02206

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/crypto_provider.h"	
Scope:	`class ara::crypto::cryp::CryptoProvider`	
Syntax:	`virtual ara::core::Result< RandomGeneratorCtx::Sptr > CreateRandomGeneratorCtx (AlgId algId) noexcept=0;`	
Parameters (in):	algId	identifier of target RNG algorithm. If no algId is given, the default RNG is returned
Return value:	ara::core::Result< RandomGenerator Ctx::Sptr >	result containing a shared pointer to the context or an error
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::k UnknownIdentifier	rollback_semantics

▽

△

		if `algId` argument has an unsupported value
Description:	Create a Random Number Generator (RNG) context.	

⌋

8.10.1.2.2.12 CreateSignerPrivateCtx

[SWS_CRYPT_20756] Definition of API function ara::crypto::cryp::CryptoProvider::CreateSignerPrivateCtx

Upstream requirements: RS_CRYPTO_02204, RS_AP_00144

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/crypto_provider.h"	
Scope:	`class ara::crypto::cryp::CryptoProvider`	
Syntax:	`virtual ara::core::Result< SignerPrivateCtx::Sptr > CreateSignerPrivateCtx (AlgId algId) noexcept=0;`	
Parameters (in):	algId	identifier of the target signature crypto algorithm
Return value:	ara::core::Result< SignerPrivateCtx::Sptr >	Smart pointer to the created context
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kInvalidArgument	rollback_semantics
		if `algId` argument specifies a crypto algorithm different from private key signature
	ara::crypto::CryptoErrc::kUnknownIdentifier	rollback_semantics
		if `algId` argument has an unsupported value
Description:	Create a signature private key context.	

⌋

8.10.1.2.2.13 CreateSymmetricCipherCtx

[SWS_CRYPT_20742] Definition of API function ara::crypto::cryp::CryptoProvider::CreateSymmetricCipherCtx

Upstream requirements: RS_CRYPTO_02201

⌈

Kind:	function
Header file:	#include "ara/crypto/cryp/crypto_provider.h"
Scope:	`class ara::crypto::cryp::CryptoProvider`
Syntax:	`virtual ara::core::Result< SymmetricCipherCtx::Sptr > CreateSymmetricCipherCtx (AlgId algId) noexcept=0;`

▽

△

Parameters (in):	algId	identifier of the target crypto algorithm
Return value:	ara::core::Result< SymmetricCipher Ctx::Sptr >	Smart pointer to the created context
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::k UnknownIdentifier	rollback_semantics
		if `AlgId` argument has an unsupported value
Description:	Create a symmetric cipher context.	

⌋

8.10.1.2.2.14 CreateSymmetricKeyWrapperCtx

[SWS_CRYPT_20743] Definition of API function ara::crypto::cryp::CryptoProvider::CreateSymmetricKeyWrapperCtx

Upstream requirements: RS_CRYPTO_02104, RS_CRYPTO_02208

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/crypto_provider.h"	
Scope:	`class ara::crypto::cryp::CryptoProvider`	
Syntax:	`virtual ara::core::Result< SymmetricKeyWrapperCtx::Sptr > CreateSymmetricKeyWrapperCtx (AlgId algId) noexcept=0;`	
Parameters (in):	algId	identifier of the target crypto algorithm
Return value:	ara::core::Result< SymmetricKeyWrapper Ctx::Sptr >	Smart pointer to the created context
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::k UnknownIdentifier	rollback_semantics
		if `algId` argument has an unsupported value
Description:	Create a symmetric key-wrap algorithm context.	

⌋

8.10.1.2.2.15 CreateVerifierPublicCtx

[SWS_CRYPT_20757] Definition of API function ara::crypto::cryp::CryptoProvider::CreateVerifierPublicCtx

Upstream requirements: RS_CRYPTO_02204, RS_AP_00144

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/crypto_provider.h"	
Scope:	`class ara::crypto::cryp::CryptoProvider`	
Syntax:	`virtual ara::core::Result< VerifierPublicCtx::Sptr > CreateVerifierPublicCtx (AlgId algId) noexcept=0;`	
Parameters (in):	algId	identifier of the target signature crypto algorithm
Return value:	ara::core::Result< VerifierPublicCtx::Sptr >	Smart pointer to the created context
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kInvalidArgument	rollback_semantics
		if `algId` argument specifies a crypto algorithm different from public key signature verification
	ara::crypto::CryptoErrc::kUnknownIdentifier	rollback_semantics
		if `algId` argument has an unsupported value
Description:	Create a signature verification public key context.	

⌋

8.10.1.2.2.16 ExportSecuredKey

[SWS_CRYPT_20728] Definition of API function ara::crypto::cryp::CryptoProvider::ExportSecuredKey

Upstream requirements: RS_CRYPTO_02105, RS_CRYPTO_02112

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/crypto_provider.h"	
Scope:	`class ara::crypto::cryp::CryptoProvider`	
Syntax:	`virtual ara::core::Result< std::size_t > ExportSecuredKey (AlgId algId, CryptoObject::Sptr key, ReadWriteMemRegion resultBuffer, ara::core::Optional< SymmetricKey::Sptr > kek) noexcept=0;`	
Parameters (in):	algId	the algId of the secure key export algorithm
	key	the key to be exported
	resultBuffer	an output buffer the securely exported key is written to
	kek	an optional key-encryption-key to be be used for wrapping the secured payload. Only used if the used algorithm does not determine the used key (e.g., based on configuration).

▽

△

Return value:	ara::core::Result< std::size_t >	a result containing the size of the result written into resultBuffer or an error
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kUnknownIdentifier	rollback_semantics
		if `algId` has an unsupported value
	ara::crypto::CryptoErrc::kInvalidArgument	rollback_semantics
		if the `key` object cannot be wrapped (e.g., invalid size or type) or if a `kek` is expected but not provided.
	ara::crypto::CryptoErrc::kUsageViolation	rollback_semantics
		if the allowed usage of `key` or `kek` is incompatible to the used algorithm
	ara::crypto::CryptoErrc::kInsufficientCapacity	rollback_semantics
		if the output buffer is too small
	ara::crypto::CryptoErrc::kBusyResource	rollback_semantics
		a required resource is busy
	ara::crypto::CryptoErrc::kIncompatibleArguments	rollback_semantics
		if the `kek` is incompatible to the used algorithm
Description:	The interface shall format the data referenced by the specified key, securely wrap this payload and serialize the secured payload into the provided output buffer. The format shall be specified implicitly by the AlgId. The formatted payload shall be wrapped according to the specified algId using the kek; if kek is not provided, ExportSecuredKey shall apply a key-encryption-key implictly defined by the algId or the default key-encryption-key of the CryptoProvider.	

⌋

8.10.1.2.2.17 GeneratePrivateKey

[SWS_CRYPT_20722] Definition of API function ara::crypto::cryp::CryptoProvider::GeneratePrivateKey

Upstream requirements: RS_CRYPTO_02003, RS_CRYPTO_02101, RS_CRYPTO_02102, RS_CRYPTO_02107, RS_CRYPTO_02108, RS_CRYPTO_02111, RS_CRYPTO_02115

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/crypto_provider.h"	
Scope:	`class ara::crypto::cryp::CryptoProvider`	
Syntax:	`virtual ara::core::Result< PrivateKey::Sptr > GeneratePrivateKey (AlgId algId, AllowedUsageFlags privateKeyAllowedUsage) noexcept=0;`	
Parameters (in):	algId	the identifier of the target crypto algorithm
	privateKeyAllowedUsage	the flags that define a list of allowed transformations' types in which the private key can be used
Return value:	ara::core::Result< PrivateKey::Sptr >	result containing a shared pointer to the created private key or an error
Exception Safety:	exception safe	
Thread Safety:	thread-safe	

▽

△

Errors:	ara::crypto::CryptoErrc::kUnknownIdentifier	rollback_semantics
		if `algId` has an unsupported value
	ara::crypto::CryptoErrc::kIncompatibleArguments	rollback_semantics
		if the `allowedUsage` is incompatible with the `algId` or the key type
	ara::crypto::CryptoErrc::kBusyResource	rollback_semantics
		if the operation temporarily cannot be performed
Description:	Generate a new public/private key pair and return the private key. The public key can be retrieved via the function `PrivateKey::GetPublicKey()` The allowed usage flags of the Public Key are set based on the allowed usage of the PrivateKey: • `kAllowDataDecryption` --> `kAllowDataEncryption` • `kAllowSigGeneration` --> `kAllowSigVerification` • `kAllowKeyAgreement` --> `kAllowKeyAgreement` • `kAllowKeyUnwrapping` --> `kAllowKeyWrapping` Additionally, the allowed usage: • `kAllowPlaintextExport` • `kAllowPersist` are set for the public key.	

⌋

8.10.1.2.2.18 GenerateRandomData

[SWS_CRYPT_30098] Definition of API function ara::crypto::cryp::CryptoProvider::GenerateRandomData

Upstream requirements: RS_CRYPTO_02206

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/crypto_provider.h"	
Scope:	`class ara::crypto::cryp::CryptoProvider`	
Syntax:	`virtual ara::core::Result< void > GenerateRandomData (ReadWriteMemRegion randomData) noexcept=0;`	
Parameters (out):	randomData	output data buffer to be filled with random data
Return value:	ara::core::Result< void >	an empty result or a result containing an error
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kBusyResource	rollback_semantics
		if the used RNG is currently out-of-entropy and therefore cannot provide the requested number of random bytes
	ara::crypto::CryptoErrc::kUnsupported	rollback_semantics
		if no default RNG is supported by the CryptoProvider
Description:	Fill the provided output data buffer randomData with random data.	

⌋

8.10.1.2.2.19 GenerateSymmetricKey

[SWS_CRYPT_20721] Definition of API function ara::crypto::cryp::CryptoProvider::GenerateSymmetricKey

Upstream requirements: RS_CRYPTO_02003, RS_CRYPTO_02101, RS_CRYPTO_02102, RS_CRYPTO_02107, RS_CRYPTO_02108, RS_CRYPTO_02111, RS_CRYPTO_02115

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/crypto_provider.h"	
Scope:	`class ara::crypto::cryp::CryptoProvider`	
Syntax:	`virtual ara::core::Result< SymmetricKey::Sptr > GenerateSymmetricKey (AlgId algId, AllowedUsageFlags allowedUsage) noexcept=0;`	
Parameters (in):	algId	the identifier of the target symmetric crypto algorithm
	allowedUsage	the flags that define a list of allowed transformations' types in which the generated key can be used
Return value:	ara::core::Result< SymmetricKey::Sptr >	result containing a shared pointer to the created symmetric key object or an error
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kUnknownIdentifier	rollback_semantics
		if `algId` has an unsupported value
	ara::crypto::CryptoErrc::kIncompatibleArguments	rollback_semantics
		if the `allowedUsage` is incompatible with the `algId` or the key type
	ara::crypto::CryptoErrc::kBusyResource	rollback_semantics
		if the operation temporarily cannot be performed
Description:	Generate a new symmetric key.	

⌋

8.10.1.2.2.20 GetDefaultAllowedUsage

[SWS_CRYPT_20955] Definition of API function ara::crypto::cryp::CryptoProvider::GetDefaultAllowedUsage

Upstream requirements: RS_CRYPTO_02003

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/crypto_provider.h"	
Scope:	`class ara::crypto::cryp::CryptoProvider`	
Syntax:	`virtual ara::core::Result< AllowedUsageFlags > GetDefaultAllowedUsage (AlgId algId) noexcept=0;`	
Parameters (in):	algId	the identifier of the target crypto algorithm

▽

△

Return value:	ara::core::Result< AllowedUsageFlags >	result containing a CryptoProvider specific default set of allowed Usages or an error
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kUnknownIdentifier	rollback_semantics
		if `algId` has an unsupported value
Description:	Get a CryptoProvider specific default set of allowedUsages for the provided AlgId. The returned allowedUsages shall not contain `kAllowPlaintextExport`, `kAllowPersist` and `kAllowSecuredExport`.	

⌋

8.10.1.2.2.21 GetProviderID

[SWS_CRYPT_50015] Definition of API function ara::crypto::cryp::CryptoProvider::GetProviderID

Upstream requirements: RS_CRYPTO_02305

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/crypto_provider.h"	
Scope:	`class ara::crypto::cryp::CryptoProvider`	
Syntax:	`virtual ara::core::Result< void > GetProviderID (ara::core::Optional< ReadOnlyMemRegion > in, ReadWriteMemRegion idData) noexcept=0;`	
Parameters (in):	in	Optional input data required by some identification protocols (e.g. SHE/CMD_GET_ID)
Parameters (out):	idData	the identification data of the provider
Return value:	ara::core::Result< void >	An empty result or an error
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kUnsupported	rollback_semantics
		if this provider does not support identification
	ara::crypto::CryptoErrc::kInsufficientCapacity	rollback_semantics
		if the provided ReadWriteMemRegion parameter idData is insufficient in size to hold the identification data
Description:	Obtain provider specific identification data.	

⌋

8.10.1.2.2.22 GetProviderIdSize

[SWS_CRYPT_50016] Definition of API function ara::crypto::cryp::CryptoProvider::GetProviderIdSize

Upstream requirements: RS_CRYPTO_02305

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/crypto_provider.h"	
Scope:	`class ara::crypto::cryp::CryptoProvider`	
Syntax:	`virtual ara::core::Result< std::size_t > GetProviderIdSize () noexcept=0;`	
Return value:	ara::core::Result< std::size_t >	a result containing the required size of ID data in bytes or an error
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kUnsupported	rollback_semantics
		if this provider does not support identification
Description:	Get the required size of the output buffer for ID data used by GetProviderID.	

⌋

8.10.1.2.2.23 GetSecureCounter

[SWS_CRYPT_20098] Definition of API function ara::crypto::cryp::CryptoProvider::GetSecureCounter

Upstream requirements: RS_CRYPTO_02401

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/crypto_provider.h"	
Scope:	`class ara::crypto::cryp::CryptoProvider`	
Syntax:	`ara::core::Result< SecureCounter > GetSecureCounter () noexcept;`	
Return value:	ara::core::Result< SecureCounter >	a result containing a SecureCounter struct made up of the two unsigned 64 bit values (LSQW and MSQW), or an error:
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kUnsupported	rollback_semantics
		if the Secure Counter is not supported by the CryptoProvider
	ara::crypto::CryptoErrc::kBusyResource	rollback_semantics
		a required resource is busy
Description:	Get the current value of the secure counter of this CryptoProvider. The current counter value is guaranteed to be stricly larger than any previously returned counter value.	

⌋

8.10.1.2.2.24 ImportPrivateKey

[SWS_CRYPT_50132] Definition of API function ara::crypto::cryp::CryptoProvider::ImportPrivateKey

Upstream requirements: RS_CRYPTO_02105, RS_CRYPTO_02112

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/crypto_provider.h"	
Scope:	`class ara::crypto::cryp::CryptoProvider`	
Syntax:	`virtual ara::core::Result< PrivateKey::Sptr > ImportPrivateKey (ReadOnlyMemRegion serialized, AllowedUsageFlags allowedUsage, FormatId formatId, ara::core::Optional< AlgId > algId) noexcept=0;`	
Parameters (in):	serialized	the memory region that contains the serialized key
	allowedUsage	the allowedUsage of the key
	formatId	the serialization format
	algId	an optional alg ID of the key. Not required if the serialization format already identifies the required properties of the key.
Return value:	ara::core::Result< PrivateKey::Sptr >	result containing a shared pointer to the created key or an error
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kInvalidArgument	rollback_semantics
		if `serialized` does not conform to the specified format
	ara::crypto::CryptoErrc::kIncompatibleArguments	rollback_semantics
		if the `allowedUsage` is incompatible with the `algId`
	ara::crypto::CryptoErrc::kUnsupported	rollback_semantics
		if the CryptoProvider does not support the requested function
	ara::crypto::CryptoErrc::kBusyResource	rollback_semantics
		if the operation temporarily cannot be performed
	ara::crypto::CryptoErrc::kUnsupportedFormat	rollback_semantics
		if the specified format ID is not supported for this object type
Description:	Import a serialized private key.	

⌋

8.10.1.2.2.25 ImportPublicKey

[SWS_CRYPT_20732] Definition of API function ara::crypto::cryp::CryptoProvider::ImportPublicKey

Upstream requirements: RS_CRYPTO_02105, RS_CRYPTO_02112

⌈

Kind:	function
Header file:	#include "ara/crypto/cryp/crypto_provider.h"
Scope:	`class ara::crypto::cryp::CryptoProvider`

▽

△

Syntax:	`virtual ara::core::Result< PublicKey::Sptr > ImportPublicKey (ReadOnlyMemRegion serialized, AllowedUsageFlags allowedUsage, FormatId formatId, ara::core::Optional< AlgId > algId) noexcept=0;`	
Parameters (in):	serialized	the memory region that contains the serialized key
	allowedUsage	the allowedUsage of the key
	formatId	the serialization format
	algId	an optional alg ID of the key. Not required if the serialization format already identifies the required properties of the key.
Return value:	ara::core::Result< PublicKey::Sptr >	result containing a shared pointer to the created key or an error
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kInvalidArgument	rollback_semantics
		if `serialized` does not conform to the specified format
	ara::crypto::CryptoErrc::kIncompatibleArguments	rollback_semantics
		if the `allowedUsage` is incompatible with the `algId`
	ara::crypto::CryptoErrc::kUnsupported	rollback_semantics
		if the CryptoProvider does not support the requested function
	ara::crypto::CryptoErrc::kBusyResource	rollback_semantics
		if the operation temporarily cannot be performed
	ara::crypto::CryptoErrc::kUnsupportedFormat	rollback_semantics
		if the specified format ID is not supported for this object type
Description:	Import a serialized public key.	

⌋

8.10.1.2.2.26 ImportSymmetricKey

[SWS_CRYPT_50014] Definition of API function ara::crypto::cryp::CryptoProvider::ImportSymmetricKey

Upstream requirements: RS_CRYPTO_02105, RS_CRYPTO_02112

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/crypto_provider.h"	
Scope:	`class ara::crypto::cryp::CryptoProvider`	
Syntax:	`virtual ara::core::Result< SymmetricKey::Sptr > ImportSymmetricKey (ReadOnlyMemRegion serialized, AllowedUsageFlags allowedUsage, FormatId formatId, ara::core::Optional< AlgId > algId) noexcept=0;`	
Parameters (in):	serialized	the memory region that contains the serialized key
	allowedUsage	the allowedUsage of the key
	formatId	the serialization format
	algId	an optional alg ID of the key. Not required if the serialization format already identifies the required properties of the key.

▽

△

Return value:	ara::core::Result< SymmetricKey::Sptr >	result containing a shared pointer to the created key or an error
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kInvalidArgument	rollback_semantics
		if `serialized` does not conform to the specified format
	ara::crypto::CryptoErrc::kIncompatibleArguments	rollback_semantics
		if the `allowedUsage` is incompatible with the `algId`
	ara::crypto::CryptoErrc::kUnsupported	rollback_semantics
		if the CryptoProvider does not support the requested function
	ara::crypto::CryptoErrc::kBusyResource	rollback_semantics
		if the operation temporarily cannot be performed
	ara::crypto::CryptoErrc::kUnsupportedFormat	rollback_semantics
		if the specified format ID is not supported for this object type
Description:	Import a serialized symmetric key.	

⌋

8.10.1.2.2.27 LoadPrivateKey

[SWS_CRYPT_20764] Definition of API function ara::crypto::cryp::CryptoProvider::LoadPrivateKey

Upstream requirements: RS_CRYPTO_02001, RS_CRYPTO_02002

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/crypto_provider.h"	
Scope:	`class ara::crypto::cryp::CryptoProvider`	
Syntax:	`virtual ara::core::Result< PrivateKey::Sptr > LoadPrivateKey (KeySlot::Sptr keySlot) noexcept=0;`	
Parameters (in):	keySlot	the KeySlot that contains the key for loading
Return value:	ara::core::Result< PrivateKey::Sptr >	result containing a shared pointer to the loaded key or an error
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kEmptyKeySlot	rollback_semantics
		if the KeySlot is empty
	ara::crypto::CryptoErrc::kIncompatibleObject	rollback_semantics
		if the underlying resource belongs to another, incompatible Crypto Provider or contains a different key type
	ara::crypto::CryptoErrc::kBusyResource	rollback_semantics
		if the operation temporarily cannot be performed
Description:	Load a private key from the provided KeySlot.	

⌋

8.10.1.2.2.28 LoadPublicKey

[SWS_CRYPT_20763] Definition of API function ara::crypto::cryp::CryptoProvider::LoadPublicKey

Upstream requirements: RS_CRYPTO_02001, RS_CRYPTO_02002

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/crypto_provider.h"	
Scope:	`class ara::crypto::cryp::CryptoProvider`	
Syntax:	`virtual ara::core::Result< PublicKey::Sptr > LoadPublicKey (KeySlot::Sptr keySlot) noexcept=0;`	
Parameters (in):	keySlot	the KeySlot that contains the key for loading
Return value:	ara::core::Result< PublicKey::Sptr >	result containing a shared pointer to the loaded key or an error
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kEmptyKeySlot	rollback_semantics
		if the KeySlot is empty
	ara::crypto::CryptoErrc::kIncompatibleObject	rollback_semantics
		if the underlying resource belongs to another, incompatible Crypto Provider or contains a different key type
	ara::crypto::CryptoErrc::kBusyResource	rollback_semantics
		if the operation temporarily cannot be performed
Description:	Load a public key from the provided KeySlot.	

⌋

8.10.1.2.2.29 LoadSymmetricKey

[SWS_CRYPT_20762] Definition of API function ara::crypto::cryp::CryptoProvider::LoadSymmetricKey

Upstream requirements: RS_CRYPTO_02001, RS_CRYPTO_02002

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/crypto_provider.h"	
Scope:	`class ara::crypto::cryp::CryptoProvider`	
Syntax:	`virtual ara::core::Result< SymmetricKey::Sptr > LoadSymmetricKey (KeySlot::Sptr keySlot) noexcept=0;`	
Parameters (in):	keySlot	the KeySlot that contains the key for loading
Return value:	ara::core::Result< SymmetricKey::Sptr >	result containing a shared pointer to the loaded key or an error
Exception Safety:	exception safe	
Thread Safety:	thread-safe	

▽

△

Errors:	ara::crypto::CryptoErrc::kEmptyKeySlot	rollback_semantics
		if the KeySlot is empty
	ara::crypto::CryptoErrc::kIncompatibleObject	rollback_semantics
		if the underlying resource belongs to another, incompatible Crypto Provider or contains a different key type
	ara::crypto::CryptoErrc::kBusyResource	rollback_semantics
		if the operation temporarily cannot be performed
Description:	Load a symmetric key from the provided KeySlot.	

⌋

8.11 Header: ara/crypto/cryp/decryptor_private_ctx.h

8.11.1 Class: DecryptorPrivateCtx

[SWS_CRYPT_50155] Definition of API class ara::crypto::cryp::DecryptorPrivateCtx

Upstream requirements: RS_CRYPTO_02201

⌈

Kind:	class
Header file:	#include "ara/crypto/cryp/decryptor_private_ctx.h"
Forwarding header file:	#include "ara/crypto/crypto_fwd.h"
Scope:	namespace ara::crypto::cryp
Symbol:	DecryptorPrivateCtx
Base class:	CryptoContext
Syntax:	`class DecryptorPrivateCtx : public CryptoContext {...};`
Description:	Interface for asymmetric decryption.

⌋

8.11.1.1 Public Member Types

8.11.1.1.1 Type Alias: Sptr

[SWS_CRYPT_50156] Definition of API type ara::crypto::cryp::DecryptorPrivateCtx::Sptr

Upstream requirements: RS_CRYPTO_02201

⌈

Kind:	type alias
Header file:	#include "ara/crypto/cryp/decryptor_private_ctx.h"
Scope:	`class ara::crypto::cryp::DecryptorPrivateCtx`
Symbol:	Sptr
Syntax:	`using Sptr = std::shared_ptr<DecryptorPrivateCtx>;`
Description:	Smart pointer of the interface.

⌋

8.11.1.2 Public Member Functions

8.11.1.2.1 Member Functions

8.11.1.2.1.1 Cancel

[SWS_CRYPT_50069] Definition of API function ara::crypto::cryp::DecryptorPrivateCtx::Cancel

Upstream requirements: RS_CRYPTO_02302

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/decryptor_private_ctx.h"	
Scope:	`class ara::crypto::cryp::DecryptorPrivateCtx`	
Syntax:	`virtual ara::core::Result< void > Cancel () noexcept=0;`	
Return value:	ara::core::Result< void >	an empty result if the ongoing streaming operation was cancelled successfully, or a result containing an error
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kProcessingNotStarted	rollback_semantics
		if data processing was not started by a call of the `Start()` method
Description:	Cancel the ongoing streaming operation that has been initiated by Start() and has not yet been finished.	

⌋

8.11.1.2.1.2 Decrypt

[SWS_CRYPT_50070] Definition of API function ara::crypto::cryp::DecryptorPrivateCtx::Decrypt

Upstream requirements: RS_CRYPTO_02302

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/decryptor_private_ctx.h"	
Scope:	`class ara::crypto::cryp::DecryptorPrivateCtx`	
Syntax:	`virtual ara::core::Result< std::size_t > Decrypt (ReadOnlyMemRegion in, ReadWriteMemRegion out) noexcept=0;`	
Parameters (in):	in	an input data buffer containing the ciphertext
Parameters (out):	out	an output data buffer that the decrypted data is written to
Return value:	ara::core::Result< std::size_t >	a result containing the size of the output data or an error. The output data may be smaller than the size of the output buffer `out`.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kUninitializedContext	rollback_semantics
		if the context was not initialized by deploying a key
	ara::crypto::CryptoErrc::kUnsupported	rollback_semantics
		if single-call decryption is not supported
	ara::crypto::CryptoErrc::kInsufficientCapacity	rollback_semantics
		if the output buffer is too small
	ara::crypto::CryptoErrc::kInvalidInputSize	rollback_semantics
		if the size of the input buffer cannot be processed
	ara::crypto::CryptoErrc::kInvalidArgument	rollback_semantics
		if the provided input data contains invalid data
	ara::crypto::CryptoErrc::kBusyResource	rollback_semantics
		a required resource is busy
	ara::crypto::CryptoErrc::kEmptyKeySlot	rollback_semantics
		if a KeySlot was configured that is currently is empty
	ara::crypto::CryptoErrc::kUsageViolation	rollback_semantics
		if the allowed usage of the key does not include kAllowDataDecryption
	ara::crypto::CryptoErrc::kIncompatibleObject	rollback_semantics
		if type of the key is incompatible to the transformation
Description:	Decrypt data in a single-call.	

⌋

8.11.1.2.1.3 Finish

[SWS_CRYPT_50068] Definition of API function ara::crypto::cryp::DecryptorPrivateCtx::Finish

Upstream requirements: RS_CRYPTO_02302

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/decryptor_private_ctx.h"	
Scope:	`class ara::crypto::cryp::DecryptorPrivateCtx`	
Syntax:	`virtual ara::core::Result< std::size_t > Finish (ReadOnlyMemRegion in, ReadWriteMemRegion out) noexcept=0;`	
Parameters (in):	in	an input data buffer containing a part of the ciphertext
Parameters (out):	out	an output data buffer that contains decrypted data. The required size of this buffer can be queried using GetRequiredOutputBuffer Size()
Return value:	ara::core::Result< std::size_t >	a result containing the size of the output data or an error. The output data may be smaller than the size of the output buffer `out`.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kProcessingNotStarted	rollback_semantics
		if data processing was not started by a call of the `Start()` method
	ara::crypto::CryptoErrc::kInsufficientCapacity	rollback_semantics
		if the output buffer is too small
	ara::crypto::CryptoErrc::kInvalidInputSize	rollback_semantics
		if the size of the input buffer cannot be processed
	ara::crypto::CryptoErrc::kInvalidArgument	rollback_semantics
		if the provided input data contains invalid data
	ara::crypto::CryptoErrc::kUnsupported	rollback_semantics
		if stream processing is not supported by the context
Description:	Finalize the decryption.	

⌋

8.11.1.2.1.4 GetMaximumChunkSize

[SWS_CRYPT_50071] Definition of API function ara::crypto::cryp::DecryptorPrivateCtx::GetMaximumChunkSize

Upstream requirements: RS_CRYPTO_02309

⌈

Kind:	function
Header file:	#include "ara/crypto/cryp/decryptor_private_ctx.h"
Scope:	`class ara::crypto::cryp::DecryptorPrivateCtx`
Syntax:	`virtual ara::core::Result< std::size_t > GetMaximumChunkSize () const noexcept=0;`

▽

△

Return value:	ara::core::Result< std::size_t >	the maximum buffer size in bytes
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	This function does not specify any standardized errors.	
Description:	Get the maximum input buffer size supported for processing in single calls to Update(), Finish(), or Decrypt() by this context.	

⌋

8.11.1.2.1.5 GetMinimumChunkSize

[SWS_CRYPT_50160] Definition of API function ara::crypto::cryp::DecryptorPrivateCtx::GetMinimumChunkSize

Upstream requirements: RS_CRYPTO_02309

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/decryptor_private_ctx.h"	
Scope:	`class ara::crypto::cryp::DecryptorPrivateCtx`	
Syntax:	`virtual ara::core::Result< std::size_t > GetMinimumChunkSize () const noexcept=0;`	
Return value:	ara::core::Result< std::size_t >	the minimum buffer size in bytes
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	This function does not specify any standardized errors.	
Description:	Get the minimum input buffer size supported for processing in single calls to Update(), Finish(), or Decrypt() by this context.	

⌋

8.11.1.2.1.6 GetRequiredOutputBufferSize

[SWS_CRYPT_50159] Definition of API function ara::crypto::cryp::DecryptorPrivateCtx::GetRequiredOutputBufferSize

Upstream requirements: RS_CRYPTO_02302

⌈

Kind:	function
Header file:	#include "ara/crypto/cryp/decryptor_private_ctx.h"
Scope:	`class ara::crypto::cryp::DecryptorPrivateCtx`
Syntax:	`virtual ara::core::Result< std::size_t > GetRequiredOutputBufferSize (std::size_t inputSize, bool isFinal) const noexcept;`

▽

△

Parameters (in):	inputSize	size of input data
	isFinal	if `true`, calculate maximum required output buffer size for Finish() or Decrypt(), otherwise for Update()
Return value:	ara::core::Result< std::size_t >	a result containing the required capacity of the output buffer (in bytes) or an error
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kUninitializedContext	rollback_semantics
		if the context was not initialized by deploying a key and setting the transformation direction
Description:	Calculate the maximum required size of the output buffer depending on the input size in the current state of the context.	

⌋

8.11.1.2.1.7 SetKey

[SWS_CRYPT_50157] Definition of API function ara::crypto::cryp::DecryptorPrivateCtx::SetKey

Upstream requirements: RS_CRYPTO_02001, RS_CRYPTO_02003

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/decryptor_private_ctx.h"	
Scope:	`class ara::crypto::cryp::DecryptorPrivateCtx`	
Syntax:	`virtual ara::core::Result< void > SetKey (PrivateKey::Sptr key) noexcept=0;`	
Parameters (in):	key	the PrivateKey to be used for decryption
Return value:	ara::core::Result< void >	An empty result or an error
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kUnsupported	rollback_semantics
		if the context only supports setting a KeySlot
	ara::crypto::CryptoErrc::kIncompatibleObject	rollback_semantics
		if the provided key object is incompatible with this context
	ara::crypto::CryptoErrc::kUsageViolation	rollback_semantics
		if the allowed usage of the key does not include kAllowDataDecryption
Description:	Configure the context to use the provided key.	

⌋

8.11.1.2.1.8 SetKey

[SWS_CRYPT_50066] Definition of API function ara::crypto::cryp::DecryptorPrivateCtx::SetKey

Upstream requirements: RS_CRYPTO_02001, RS_CRYPTO_02003

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/decryptor_private_ctx.h"	
Scope:	`class ara::crypto::cryp::DecryptorPrivateCtx`	
Syntax:	`virtual ara::core::Result< void > SetKey (KeySlot::Sptr keySlot) noexcept=0;`	
Parameters (in):	keySlot	a keyslot containing a PrivateKey.
Return value:	ara::core::Result< void >	An empty result or an error
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kIncompatibleObject	rollback_semantics
		if the provided keyslot is incompatible with this context
	ara::crypto::CryptoErrc::kUnsupported	rollback_semantics
		if the context only supports setting a PrivateKey
Description:	Configure the context to use the key stored in the provided KeySlot at the time the decryption is started.	

⌋

8.11.1.2.1.9 Start

[SWS_CRYPT_50067] Definition of API function ara::crypto::cryp::DecryptorPrivateCtx::Start

Upstream requirements: RS_CRYPTO_02302

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/decryptor_private_ctx.h"	
Scope:	`class ara::crypto::cryp::DecryptorPrivateCtx`	
Syntax:	`virtual ara::core::Result< void > Start () noexcept=0;`	
Return value:	ara::core::Result< void >	An empty result or an error
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kUninitializedContext	rollback_semantics
		if the context was not initialized by deploying a key
	ara::crypto::CryptoErrc::kBusyResource	rollback_semantics
		a required resource is busy
	ara::crypto::CryptoErrc::kEmptyKeySlot	rollback_semantics
		if a KeySlot was configured that is currently is empty

▽

△

	ara::crypto::CryptoErrc::kUsageViolation	rollback_semantics
		if the allowed usage of the key does not include kAllowDataDecryption
	ara::crypto::CryptoErrc::kIncompatibleObject	rollback_semantics
		if type of the key is incompatible to the transformation
	ara::crypto::CryptoErrc::kUnsupported	rollback_semantics
		if stream processing is not supported by the context
Description:	Initialize the context for a new decryption.	

⌋

8.11.1.2.1.10 Update

[SWS_CRYPT_50158] Definition of API function ara::crypto::cryp::DecryptorPrivateCtx::Update

Upstream requirements: RS_CRYPTO_02302

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/decryptor_private_ctx.h"	
Scope:	`class ara::crypto::cryp::DecryptorPrivateCtx`	
Syntax:	`virtual ara::core::Result< std::size_t > Update (ReadOnlyMemRegion in, ReadWriteMemRegion out) noexcept=0;`	
Parameters (in):	in	an input data buffer containing a part of the ciphertext
Parameters (out):	out	an output data buffer that the decrypted data will be written into. The required size of this buffer can be queried using GetRequiredOutputBufferSize()
Return value:	ara::core::Result< std::size_t >	a result containing the size of the output data or an error. The output data may be smaller than the size of the output buffer out.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kProcessingNotStarted	rollback_semantics
		if data processing was not started by a call of the `Start()` method
	ara::crypto::CryptoErrc::kInsufficientCapacity	rollback_semantics
		if the output buffer is too small
	ara::crypto::CryptoErrc::kInvalidInputSize	rollback_semantics
		if the size of the input buffer is invalid
	ara::crypto::CryptoErrc::kUnsupported	rollback_semantics
		if stream processing is not supported by the context
Description:	Decrypt a data buffer.	

⌋

8.12 Header: ara/crypto/cryp/encryptor_public_ctx.h

8.12.1 Class: EncryptorPublicCtx

[SWS_CRYPT_50173] Definition of API class ara::crypto::cryp::EncryptorPublicCtx

Upstream requirements: RS_CRYPTO_02201

⌈

Kind:	class
Header file:	#include "ara/crypto/cryp/encryptor_public_ctx.h"
Forwarding header file:	#include "ara/crypto/crypto_fwd.h"
Scope:	namespace ara::crypto::cryp
Symbol:	EncryptorPublicCtx
Base class:	CryptoContext
Syntax:	`class EncryptorPublicCtx : public CryptoContext {...};`
Description:	Interface for asymmetric encryption.

⌋

8.12.1.1 Public Member Types

8.12.1.1.1 Type Alias: Sptr

[SWS_CRYPT_50174] Definition of API type ara::crypto::cryp::EncryptorPublicCtx::Sptr

Upstream requirements: RS_CRYPTO_02201

⌈

Kind:	type alias
Header file:	#include "ara/crypto/cryp/encryptor_public_ctx.h"
Scope:	`class ara::crypto::cryp::EncryptorPublicCtx`
Symbol:	Sptr
Syntax:	`using Sptr = std::shared_ptr<EncryptorPublicCtx>;`
Description:	Smart pointer of the interface.

⌋

8.12.1.2 Public Member Functions

8.12.1.2.1 Member Functions

8.12.1.2.1.1 Cancel

[SWS_CRYPT_50089] Definition of API function ara::crypto::cryp::EncryptorPublicCtx::Cancel

Upstream requirements: RS_CRYPTO_02302

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/encryptor_public_ctx.h"	
Scope:	`class ara::crypto::cryp::EncryptorPublicCtx`	
Syntax:	`virtual ara::core::Result< void > Cancel () noexcept=0;`	
Return value:	ara::core::Result< void >	an empty result if the ongoing streaming operation was cancelled successfully, or a result containing an error
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kProcessingNotStarted	rollback_semantics
		if data processing was not started by a call of the `Start()` method
Description:	Cancel the ongoing streaming operation that has been initiated by Start() and has not yet been finished.	

⌋

8.12.1.2.1.2 Encrypt

[SWS_CRYPT_50090] Definition of API function ara::crypto::cryp::EncryptorPublicCtx::Encrypt

Upstream requirements: RS_CRYPTO_02302

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/encryptor_public_ctx.h"	
Scope:	`class ara::crypto::cryp::EncryptorPublicCtx`	
Syntax:	`virtual ara::core::Result< std::size_t > Encrypt (ReadOnlyMemRegion in, ReadWriteMemRegion out) noexcept=0;`	
Parameters (in):	in	an input data buffer containing the plaintext
Parameters (out):	out	an output data buffer that the encrypted data is written to
Return value:	ara::core::Result< std::size_t >	a result containing the size of the output data or an error. The output data may be smaller than the size of the output buffer `out`.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kUninitializedContext	rollback_semantics
		if the context was not initialized by deploying a key

▽

△

	ara::crypto::CryptoErrc::kUnsupported	rollback_semantics
		if single-call encryption is not supported
	ara::crypto::CryptoErrc::kInsufficientCapacity	rollback_semantics
		if the output buffer is too small
	ara::crypto::CryptoErrc::kInvalidInputSize	rollback_semantics
		if the size of the input buffer cannot be processed
	ara::crypto::CryptoErrc::kInvalidArgument	rollback_semantics
		if the provided input data contains invalid data
	ara::crypto::CryptoErrc::kBusyResource	rollback_semantics
		a required resource is busy
	ara::crypto::CryptoErrc::kEmptyKeySlot	rollback_semantics
		if a KeySlot was configured that is currently is empty
	ara::crypto::CryptoErrc::kUsageViolation	rollback_semantics
		if the allowed usage of the key does not include kAllowDataEncryption
	ara::crypto::CryptoErrc::kIncompatibleObject	rollback_semantics
		if type of the key is incompatible to the transformation
Description:	Encrypt data in a single-call.	

⌋

8.12.1.2.1.3 Finish

[SWS_CRYPT_50088] Definition of API function ara::crypto::cryp::EncryptorPublicCtx::Finish

Upstream requirements: RS_CRYPTO_02302

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/encryptor_public_ctx.h"	
Scope:	`class ara::crypto::cryp::EncryptorPublicCtx`	
Syntax:	`virtual ara::core::Result< std::size_t > Finish (ReadOnlyMemRegion in, ReadWriteMemRegion out) noexcept=0;`	
Parameters (in):	in	an input data buffer containing a part of the plaintext
Parameters (out):	out	an output data buffer that contains encrypted data. The required size of this buffer can be queried using GetRequiredOutputBuffer Size()
Return value:	ara::core::Result< std::size_t >	a result containing the size of the output data or an error. The output data may be smaller than the size of the output buffer `out`.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kProcessingNotStarted	rollback_semantics
		if data processing was not started by a call of the `Start()` method
	ara::crypto::CryptoErrc::kInsufficientCapacity	rollback_semantics
		if the output buffer is too small

▽

△

	ara::crypto::CryptoErrc::kInvalidInputSize	rollback_semantics
		if the size of the input buffer cannot be processed
	ara::crypto::CryptoErrc::kInvalidArgument	rollback_semantics
		if the provided input data contains invalid data
	ara::crypto::CryptoErrc::kUnsupported	rollback_semantics
		if stream processing is not supported by the context
Description:	Finalize the encryption.	

⌋

8.12.1.2.1.4 GetMaximumChunkSize

[SWS_CRYPT_50091] Definition of API function ara::crypto::cryp::EncryptorPublicCtx::GetMaximumChunkSize

Upstream requirements: RS_CRYPTO_02309

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/encryptor_public_ctx.h"	
Scope:	`class ara::crypto::cryp::EncryptorPublicCtx`	
Syntax:	`virtual ara::core::Result< std::size_t > GetMaximumChunkSize () const noexcept=0;`	
Return value:	ara::core::Result< std::size_t >	the maximum buffer size in bytes
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	This function does not specify any standardized errors.	
Description:	Get the maximum input buffer size supported for processing in single calls to Update(), Finish(), or Encrypt() by this context.	

⌋

8.12.1.2.1.5 GetMinimumChunkSize

[SWS_CRYPT_50178] Definition of API function ara::crypto::cryp::EncryptorPublicCtx::GetMinimumChunkSize

Upstream requirements: RS_CRYPTO_02309

⌈

Kind:	function
Header file:	#include "ara/crypto/cryp/encryptor_public_ctx.h"
Scope:	`class ara::crypto::cryp::EncryptorPublicCtx`

▽

△

Syntax:	`virtual ara::core::Result< std::size_t > GetMinimumChunkSize () const noexcept=0;`	
Return value:	ara::core::Result< std::size_t >	the minimum buffer size in bytes
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	This function does not specify any standardized errors.	
Description:	Get the minimum input buffer size supported for processing in single calls to Update(), Finish(), or Encrypt() by this context.	

⌋

8.12.1.2.1.6 GetRequiredOutputBufferSize

[SWS_CRYPT_50177] Definition of API function ara::crypto::cryp::EncryptorPublicCtx::GetRequiredOutputBufferSize

Upstream requirements: RS_CRYPTO_02302

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/encryptor_public_ctx.h"	
Scope:	`class ara::crypto::cryp::EncryptorPublicCtx`	
Syntax:	`virtual ara::core::Result< std::size_t > GetRequiredOutputBufferSize (std::size_t inputSize, bool isFinal) const noexcept;`	
Parameters (in):	inputSize	size of input data
	isFinal	if `true`, calculate maximum required output buffer size for Finish() or Encrypt(), otherwise for Update()
Return value:	ara::core::Result< std::size_t >	a result containing the required capacity of the output buffer (in bytes) or an error
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kUninitializedContext	rollback_semantics
		if the context was not initialized by deploying a key and setting the transformation direction
Description:	Calculate the maximum required size of the output buffer depending on the input size in the current state of the context.	

⌋

8.12.1.2.1.7 SetKey

[SWS_CRYPT_50175] Definition of API function ara::crypto::cryp::EncryptorPublicCtx::SetKey

Upstream requirements: RS_CRYPTO_02001, RS_CRYPTO_02003

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/encryptor_public_ctx.h"	
Scope:	`class ara::crypto::cryp::EncryptorPublicCtx`	
Syntax:	`virtual ara::core::Result< void > SetKey (PublicKey::Sptr key) noexcept=0;`	
Parameters (in):	key	the PublicKey to be used for encryption
Return value:	ara::core::Result< void >	An empty result or an error
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kUnsupported	rollback_semantics
		if the context only supports setting a KeySlot
	ara::crypto::CryptoErrc::kIncompatibleObject	rollback_semantics
		if the provided key object is incompatible with this context
	ara::crypto::CryptoErrc::kUsageViolation	rollback_semantics
		if the allowed usage of the key does not include kAllowDataEncryption
Description:	Configure the context to use the provided key.	

⌋

8.12.1.2.1.8 SetKey

[SWS_CRYPT_50086] Definition of API function ara::crypto::cryp::EncryptorPublicCtx::SetKey

Upstream requirements: RS_CRYPTO_02001, RS_CRYPTO_02003

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/encryptor_public_ctx.h"	
Scope:	`class ara::crypto::cryp::EncryptorPublicCtx`	
Syntax:	`virtual ara::core::Result< void > SetKey (KeySlot::Sptr keySlot) noexcept=0;`	
Parameters (in):	keySlot	a keyslot containing a PublicKey.
Return value:	ara::core::Result< void >	An empty result or an error
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kIncompatibleObject	rollback_semantics
		if the provided keyslot is incompatible with this context

▽

△

	ara::crypto::CryptoErrc::kUnsupported	rollback_semantics
		if the context only supports setting a PublicKey
Description:	Configure the context to use the key stored in the provided KeySlot at the time the encryption is started.	

⌋

8.12.1.2.1.9 Start

[SWS_CRYPT_50087] Definition of API function ara::crypto::cryp::EncryptorPublicCtx::Start

Upstream requirements: RS_CRYPTO_02302

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/encryptor_public_ctx.h"	
Scope:	`class ara::crypto::cryp::EncryptorPublicCtx`	
Syntax:	`virtual ara::core::Result< void > Start () noexcept=0;`	
Return value:	ara::core::Result< void >	An empty result or an error
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kUninitializedContext	rollback_semantics
		if the context was not initialized by deploying a key
	ara::crypto::CryptoErrc::kBusyResource	rollback_semantics
		a required resource is busy
	ara::crypto::CryptoErrc::kEmptyKeySlot	rollback_semantics
		if a KeySlot was configured that is currently is empty
	ara::crypto::CryptoErrc::kUsageViolation	rollback_semantics
		if the allowed usage of the key does not include kAllowDataEncryption
	ara::crypto::CryptoErrc::kIncompatibleObject	rollback_semantics
		if type of the key is incompatible to the transformation
	ara::crypto::CryptoErrc::kUnsupported	rollback_semantics
		if stream processing is not supported by the context
Description:	Initialize the context for a new encryption.	

⌋

8.12.1.2.1.10 Update

[SWS_CRYPT_50176] Definition of API function ara::crypto::cryp::EncryptorPublicCtx::Update

Upstream requirements: RS_CRYPTO_02302

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/encryptor_public_ctx.h"	
Scope:	`class ara::crypto::cryp::EncryptorPublicCtx`	
Syntax:	`virtual ara::core::Result< std::size_t > Update (ReadOnlyMemRegion in, ReadWriteMemRegion out) noexcept=0;`	
Parameters (in):	in	an input data buffer containing a part of the plaintext
Parameters (out):	out	an output data buffer that the encrypted data will be written into. The required size of this buffer can be queried using GetRequiredOutputBufferSize()
Return value:	ara::core::Result< std::size_t >	a result containing the size of the output data or an error. The output data may be smaller than the size of the output buffer out.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kProcessingNotStarted	rollback_semantics
		if data processing was not started by a call of the `Start()` method
	ara::crypto::CryptoErrc::kInsufficientCapacity	rollback_semantics
		if the output buffer is too small
	ara::crypto::CryptoErrc::kInvalidInputSize	rollback_semantics
		if the size of the input buffer is invalid
	ara::crypto::CryptoErrc::kUnsupported	rollback_semantics
		if stream processing is not supported by the context
Description:	Encrypt a data buffer.	

⌋

8.13 Header: ara/crypto/cryp/hash_function_ctx.h

8.13.1 Class: HashFunctionCtx

[SWS_CRYPT_21100] Definition of API class ara::crypto::cryp::HashFunctionCtx

Upstream requirements: RS_CRYPTO_02205

⌈

Kind:	class
Header file:	#include "ara/crypto/cryp/hash_function_ctx.h"
Forwarding header file:	#include "ara/crypto/crypto_fwd.h"
Scope:	namespace ara::crypto::cryp
Symbol:	HashFunctionCtx

▽

△

Base class:	CryptoContext
Syntax:	`class HashFunctionCtx : public CryptoContext {...};`
Description:	Hash function interface.

⌋

8.13.1.1 Public Member Types

8.13.1.1.1 Type Alias: Sptr

[SWS_CRYPT_21101] Definition of API type ara::crypto::cryp::HashFunctionCtx::Sptr

Upstream requirements: RS_CRYPTO_02205

⌈

Kind:	type alias
Header file:	#include "ara/crypto/cryp/hash_function_ctx.h"
Scope:	`class ara::crypto::cryp::HashFunctionCtx`
Symbol:	Sptr
Syntax:	`using Sptr = std::shared_ptr<HashFunctionCtx>;`
Description:	Smart pointer of the interface.

⌋

8.13.1.2 Public Member Functions

8.13.1.2.1 Member Functions

8.13.1.2.1.1 CalculateHashDigest

[SWS_CRYPT_50060] Definition of API function ara::crypto::cryp::HashFunctionCtx::CalculateHashDigest

Upstream requirements: RS_CRYPTO_02205

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/hash_function_ctx.h"	
Scope:	`class ara::crypto::cryp::HashFunctionCtx`	
Syntax:	`virtual ara::core::Result< std::size_t > CalculateHashDigest (ReadOnly MemRegion data, ReadWriteMemRegion digest, ara::core::Optional< Read OnlyMemRegion > iv) const noexcept=0;`	
Parameters (in):	data	the input message that should be processed
	iv	an optional initialization vector

▽

△

Parameters (out):	digest	an output data buffer that the digest will be written into.
Return value:	ara::core::Result< std::size_t >	A result containing the size in bytes of the digest or an error.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kInvalidInputSize	rollback_semantics
		if the size of provided data or IV is not supported
	ara::crypto::CryptoErrc::kUnsupported	rollback_semantics
		if single-call operation is not supported, or if the context requires setting an IV but no IV is provided, or if the context does not support setting an IV but an IV is provided
	ara::crypto::CryptoErrc::kInsufficientCapacity	rollback_semantics
		if the output buffer is too small
Description:	Calculate the hash digest and write the the calculated digest into the provided output buffer.	

⌋

8.13.1.2.1.2 Cancel

[SWS_CRYPT_50058] Definition of API function ara::crypto::cryp::HashFunctionCtx::Cancel

Upstream requirements: RS_CRYPTO_02302

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/hash_function_ctx.h"	
Scope:	`class ara::crypto::cryp::HashFunctionCtx`	
Syntax:	`virtual ara::core::Result< void > Cancel () noexcept=0;`	
Return value:	ara::core::Result< void >	an empty result if the ongoing streaming operation was cancelled successfully, or a result containing an error
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kProcessingNotStarted	rollback_semantics
		if data processing was not started by a call of the `Start()` method
Description:	Cancel the ongoing streaming operation that has been initiated by Start() and has not yet been finished.	

⌋

8.13.1.2.1.3 Compare

[SWS_CRYPT_50154] Definition of API function ara::crypto::cryp::HashFunctionCtx::Compare

Upstream requirements: RS_CRYPTO_02309

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/hash_function_ctx.h"	
Scope:	`class ara::crypto::cryp::HashFunctionCtx`	
Syntax:	`virtual ara::core::Result< bool > Compare (ReadOnlyMemRegion expected) const noexcept=0;`	
Parameters (in):	expected	the memory region containing an expected digest value
Return value:	ara::core::Result< bool >	`true` if the expected byte sequence is identical to the calculated digest
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kProcessingNotFinished	rollback_semantics
		if the digest calculation was not finished by a call of the `Finish()` method
Description:	Compare the size and content of the calculated digest against an expected value.	

⌋

8.13.1.2.1.4 ExportContextState

[SWS_CRYPT_50064] Definition of API function ara::crypto::cryp::HashFunctionCtx::ExportContextState

Upstream requirements: RS_CRYPTO_02309

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/hash_function_ctx.h"	
Scope:	`class ara::crypto::cryp::HashFunctionCtx`	
Syntax:	`virtual ara::core::Result< void > ExportContextState (ReadWriteMemRegion contextState) const noexcept=0;`	
Parameters (out):	contextState	an output data buffer that the state of the context will be written into.
Return value:	ara::core::Result< void >	An empty result on success or an error
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kUnsupported	rollback_semantics
		if this context does not support im-/exporting the context state
	ara::crypto::CryptoErrc::kInsufficientCapacity	rollback_semantics
		if the output buffer is too small

▽

△

Description:	Write the current state of this hash context into the provided output buffer contextState. This function can be used to suspend an ongoing hash operation and resume it at a later point in time.

⌋

8.13.1.2.1.5 Finish

[SWS_CRYPT_21115] Definition of API function ara::crypto::cryp::HashFunctionCtx::Finish

Upstream requirements: RS_CRYPTO_02302, RS_CRYPTO_02205

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/hash_function_ctx.h"	
Scope:	`class ara::crypto::cryp::HashFunctionCtx`	
Syntax:	`virtual ara::core::Result< void > Finish () noexcept=0;`	
Return value:	ara::core::Result< void >	An empty result or an error
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kInvalidUsageOrder	rollback_semantics
		the hash calculation has not been started successfully.
Description:	Finish the digest calculation.	

⌋

8.13.1.2.1.6 GetContextStateSize

[SWS_CRYPT_50063] Definition of API function ara::crypto::cryp::HashFunctionCtx::GetContextStateSize

Upstream requirements: RS_CRYPTO_02309

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/hash_function_ctx.h"	
Scope:	`class ara::crypto::cryp::HashFunctionCtx`	
Syntax:	`virtual ara::core::Result< std::size_t > GetContextStateSize () const noexcept=0;`	
Return value:	ara::core::Result< std::size_t >	A result containing required context state buffer size in bytes or an error
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kUnsupported	rollback_semantics
		if this context does not support im-/exporting the context state

▽

△

Description:	Get the buffer size required to store the state of this context using ExportContextState()

⌋

8.13.1.2.1.7 GetDigest

[SWS_CRYPT_21116] Definition of API function ara::crypto::cryp::HashFunctionCtx::GetDigest

Upstream requirements: RS_CRYPTO_02205

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/hash_function_ctx.h"	
Scope:	`class ara::crypto::cryp::HashFunctionCtx`	
Syntax:	`virtual ara::core::Result< std::size_t > GetDigest (ReadWriteMemRegion digest) const noexcept=0;`	
Parameters (out):	digest	an output data buffer that the hash digest will be written into
Return value:	ara::core::Result< std::size_t >	A result containing the size in bytes of the hash digest or an error.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kProcessingNotFinished	rollback_semantics
		if the hash calculation was not finished by a call of the `Finish()` method
	ara::crypto::CryptoErrc::kInsufficientCapacity	rollback_semantics
		if the output buffer is too small
Description:	Write the calculated hash digest into the provided output buffer.	

⌋

8.13.1.2.1.8 GetDigestSize

[SWS_CRYPT_50061] Definition of API function ara::crypto::cryp::HashFunctionCtx::GetDigestSize

Upstream requirements: RS_CRYPTO_02309

⌈

Kind:	function
Header file:	#include "ara/crypto/cryp/hash_function_ctx.h"
Scope:	`class ara::crypto::cryp::HashFunctionCtx`
Syntax:	`virtual ara::core::Result< std::size_t > GetDigestSize () const noexcept=0;`

▽

△

Return value:	ara::core::Result< std::size_t >	A result containing the size of the output from this digest-function in bytes or an error
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	This function does not specify any standardized errors.	
Description:	Get the output digest size.	

⌋

8.13.1.2.1.9 GetIntermediateDigest

[SWS_CRYPT_50059] Definition of API function ara::crypto::cryp::HashFunctionCtx::GetIntermediateDigest

Upstream requirements: RS_CRYPTO_02205

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/hash_function_ctx.h"	
Scope:	`class ara::crypto::cryp::HashFunctionCtx`	
Syntax:	`virtual ara::core::Result< std::size_t > GetIntermediateDigest (ReadWriteMemRegion digest) const noexcept=0;`	
Parameters (out):	digest	an output data buffer that the hash digest will be written into
Return value:	ara::core::Result< std::size_t >	A result containing the size in bytes of the hash digest or an error.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kInvalidUsageOrder	rollback_semantics
		the hash calculation has not been started successfully.
	ara::crypto::CryptoErrc::kInsufficientCapacity	rollback_semantics
		if the output buffer is too small
	ara::crypto::CryptoErrc::kUnsupported	rollback_semantics
		if this context does not support getting an intemediate digest
Description:	Calculate the hash digest with the data provided to the hash context and write the calculated hash digest into the provided output buffer. The hash digest is equal to the hash digest returned when calling Finish() and GetDigest(), however, after calling GetIntermediateDigest(), this context can still be updated with additional data.	

⌋

8.13.1.2.1.10 GetMaximumChunkSize

[SWS_CRYPT_50062] Definition of API function ara::crypto::cryp::HashFunctionCtx::GetMaximumChunkSize

Upstream requirements: RS_CRYPTO_02309

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/hash_function_ctx.h"	
Scope:	`class ara::crypto::cryp::HashFunctionCtx`	
Syntax:	`virtual ara::core::Result< std::size_t > GetMaximumChunkSize () const noexcept=0;`	
Return value:	ara::core::Result< std::size_t >	A result containing the maximum buffer size in bytes or an error
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	This function does not specify any standardized errors.	
Description:	Get the maximum input buffer size supported for processing in single calls to Update(), Finish(), or CalculateHashDigest() by this context.	

⌋

8.13.1.2.1.11 ImportContextState

[SWS_CRYPT_50065] Definition of API function ara::crypto::cryp::HashFunctionCtx::ImportContextState

Upstream requirements: RS_CRYPTO_02309

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/hash_function_ctx.h"	
Scope:	`class ara::crypto::cryp::HashFunctionCtx`	
Syntax:	`virtual ara::core::Result< void > ImportContextState (ReadOnlyMemRegion contextState) noexcept=0;`	
Parameters (in):	contextState	a data buffer that contains a previously exported state of the context
Return value:	ara::core::Result< void >	An empty result on success or an error
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kUnsupported	rollback_semantics
		if this context does not support im-/exporting the context state
	ara::crypto::CryptoErrc::kInvalidArgument	rollback_semantics
		if the `contextState` argument contains invalid data

▽

△

Description:	Set the current state of this hash context to the state provided by contextState. This function can be used to resume a previously suspended hash operation. The contextState must have been exported from a HashFunctionCtx with the same AlgorithmId provided by the same Crypto Provider as the current context using the function ExportContextState(). The current state of this context (e.g., any ongoing hash calculation) will be replaced by the state provided to this function.

⌋

8.13.1.2.1.12 Start

[SWS_CRYPT_50152] Definition of API function ara::crypto::cryp::HashFunction Ctx::Start

Upstream requirements: RS_CRYPTO_02302

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/hash_function_ctx.h"	
Scope:	`class ara::crypto::cryp::HashFunctionCtx`	
Syntax:	`virtual ara::core::Result< void > Start () noexcept=0;`	
Return value:	ara::core::Result< void >	An empty result or an error
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kUnsupported	rollback_semantics
		the used hash function expects an IV
	ara::crypto::CryptoErrc::kBusyResource	rollback_semantics
		a required resource is busy
Description:	Initialize the context for processing of a hash digest without an IV. Will cancel any ongoing hash calculation.	

⌋

8.13.1.2.1.13 Start

[SWS_CRYPT_50153] Definition of API function ara::crypto::cryp::HashFunction Ctx::Start

Upstream requirements: RS_CRYPTO_02302

⌈

Kind:	function
Header file:	#include "ara/crypto/cryp/hash_function_ctx.h"
Scope:	`class ara::crypto::cryp::HashFunctionCtx`
Syntax:	`virtual ara::core::Result< void > Start (ReadOnlyMemRegion iv) noexcept=0;`

▽

△

Parameters (in):	iv	the used Initialization Vector (IV)
Return value:	ara::core::Result< void >	An empty result or an error
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kInvalidInputSize	rollback_semantics
		the size of provided IV is not supported
	ara::crypto::CryptoErrc::kUnsupported	rollback_semantics
		the used hash function does not use an IV
	ara::crypto::CryptoErrc::kBusyResource	rollback_semantics
		a required resource is busy
Description:	Initialize the context for processing of a hash digest with an IV. Will cancel any ongoing hash calculation.	

⌋

8.13.1.2.1.14 Update

[SWS_CRYPT_21113] Definition of API function ara::crypto::cryp::HashFunctionCtx::Update

Upstream requirements: RS_CRYPTO_02302

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/hash_function_ctx.h"	
Scope:	`class ara::crypto::cryp::HashFunctionCtx`	
Syntax:	`virtual ara::core::Result< void > Update (ReadOnlyMemRegion data) noexcept=0;`	
Parameters (in):	data	a part of the input message that should be processed
Return value:	ara::core::Result< void >	An empty result or an error
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kProcessingNotStarted	rollback_semantics
		if the digest calculation was not initiated by a call of the `Start()` method
	ara::crypto::CryptoErrc::kInvalidInputSize	rollback_semantics
		if the size of provided data is not supported
Description:	Update the digest calculation context by a new part of the message.	

⌋

8.13.1.2.1.15 Update

[SWS_CRYPT_50057] Definition of API function ara::crypto::cryp::HashFunctionCtx::Update

Upstream requirements: RS_CRYPTO_02302

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/hash_function_ctx.h"	
Scope:	`class ara::crypto::cryp::HashFunctionCtx`	
Syntax:	`virtual ara::core::Result< void > Update (CryptoObject::Sptr key) noexcept=0;`	
Parameters (in):	key	a part of the input message that should be processed
Return value:	ara::core::Result< void >	An empty result or an error
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kProcessingNotStarted	rollback_semantics
		if the digest calculation was not initiated by a call of the `Start()` method
	ara::crypto::CryptoErrc::kInvalidInputSize	rollback_semantics
		if the size of provided data is not supported
	ara::crypto::CryptoErrc::kUnsupported	rollback_semantics
		if this context does not support hashing of keys
	ara::crypto::CryptoErrc::kIncompatibleObject	rollback_semantics
		if the provided key is incompatible with this context
	ara::crypto::CryptoErrc::kUsageViolation	rollback_semantics
		if the allowed usage of the key does not include kAllowHashCalculation
Description:	Update the digest calculation context by a new part of the message.	

⌋

8.14 Header: ara/crypto/cryp/key_agreement_private_ctx.h

8.14.1 Class: KeyAgreementPrivateCtx

[SWS_CRYPT_21300] Definition of API class ara::crypto::cryp::KeyAgreementPrivateCtx

Upstream requirements: RS_CRYPTO_02104

⌈

Kind:	class
Header file:	#include "ara/crypto/cryp/key_agreement_private_ctx.h"
Forwarding header file:	#include "ara/crypto/crypto_fwd.h"
Scope:	namespace ara::crypto::cryp

▽

△

Symbol:	KeyAgreementPrivateCtx
Base class:	CryptoContext
Syntax:	`class KeyAgreementPrivateCtx : public CryptoContext {...};`
Description:	Key Agreement Private key Context interface.

⌋

8.14.1.1 Public Member Types

8.14.1.1.1 Type Alias: Sptr

[SWS_CRYPT_21301] Definition of API type ara::crypto::cryp::KeyAgreementPrivateCtx::Sptr

Upstream requirements: RS_CRYPTO_02104

⌈

Kind:	type alias
Header file:	#include "ara/crypto/cryp/key_agreement_private_ctx.h"
Scope:	`class ara::crypto::cryp::KeyAgreementPrivateCtx`
Symbol:	Sptr
Syntax:	`using Sptr = std::shared_ptr<KeyAgreementPrivateCtx>;`
Description:	Smart pointer of this interface.

⌋

8.14.1.2 Public Member Functions

8.14.1.2.1 Member Functions

8.14.1.2.1.1 AgreeKey

[SWS_CRYPT_21312] Definition of API function ara::crypto::cryp::KeyAgreementPrivateCtx::AgreeKey

Upstream requirements: RS_CRYPTO_02115

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/key_agreement_private_ctx.h"	
Scope:	`class ara::crypto::cryp::KeyAgreementPrivateCtx`	
Syntax:	`virtual ara::core::Result< SymmetricKey::Sptr > AgreeKey (PublicKey::Sptr otherSideKey, AlgId targetAlgId, AllowedUsageFlags allowedUsage) const noexcept=0;`	
Parameters (in):	otherSideKey	the public key of the other side of the Key-Agreement

▽

△

	targetAlgId	identifier of the symmetric target algorithm
	allowedUsage	the allowed usage of the target key
Return value:	ara::core::Result< SymmetricKey::Sptr >	a result containing either a unique pointer to the `SymmetricKey` object, which contains the computed shared secret produced by the Key-Agreement algorithm or an error
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	CryptoErrc::k UninitializedContext	rollback_semantics
		if the context was not initialized by deploying a key or not all required parameters have been set via SetParameter()
	CryptoErrc::k IncompatibleObject	rollback_semantics
		if the public key is not compatible to this context
	CryptoErrc::kInvalid Argument	rollback_semantics
		if the allowed usage flags are invalid for a symmetric key
	CryptoErrc::kBusy Resource	rollback_semantics
		a required resource is busy
	CryptoErrc::kEmptyKey Slot	rollback_semantics
		if a KeySlot was configured that is currently is empty
	CryptoErrc::kUsage Violation	rollback_semantics
		if the allowed usage of the private or public key do not include k AllowKeyAgreement
	ara::crypto::CryptoErrc::k UnknownIdentifier	rollback_semantics
		if `targetAlgId` has an unsupported value
Description:	Produce a common symmetric key via execution of the key-agreement algorithm using a private key and a public key of another side.	

⌋

8.14.1.2.1.2 SetKey

[SWS_CRYPT_50053] Definition of API function ara::crypto::cryp::KeyAgreementPrivateCtx::SetKey

Upstream requirements: RS_CRYPTO_02001, RS_CRYPTO_02003

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/key_agreement_private_ctx.h"	
Scope:	`class ara::crypto::cryp::KeyAgreementPrivateCtx`	
Syntax:	`virtual ara::core::Result< void > SetKey (KeySlot::Sptr keySlot) noexcept=0;`	
Parameters (in):	keySlot	a keyslot containing a private key
Return value:	ara::core::Result< void >	An empty result or an error
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::k IncompatibleObject	rollback_semantics

▽

△

		if the provided keySlot is incompatible with this context
	ara::crypto::CryptoErrc::kUnsupported	rollback_semantics
		if the context only supports setting a PrivateKey
Description:	Configure the context to use the key stored in the provided KeySlot at the time the key agreement is performed.	

⌋

8.14.1.2.1.3 SetKey

[SWS_CRYPT_21313] Definition of API function ara::crypto::cryp::KeyAgreementPrivateCtx::SetKey

Upstream requirements: RS_CRYPTO_02001, RS_CRYPTO_02003

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/key_agreement_private_ctx.h"	
Scope:	`class ara::crypto::cryp::KeyAgreementPrivateCtx`	
Syntax:	`virtual ara::core::Result< void > SetKey (PrivateKey::Sptr key) noexcept=0;`	
Parameters (in):	key	the source key object
Return value:	ara::core::Result< void >	An empty result or an error
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kIncompatibleObject	rollback_semantics
		if the provided key object is incompatible with this private key context
	ara::crypto::CryptoErrc::kUsageViolation	rollback_semantics
		if the allowed usage of the key does not include kAllowKeyAgreement
	ara::crypto::CryptoErrc::kUnsupported	rollback_semantics
		if the context only supports setting a KeySlot
Description:	Configure the context to use the provided key.	

⌋

8.14.1.2.1.4 SetParameter

[SWS_CRYPT_50051] Definition of API function ara::crypto::cryp::KeyAgreementPrivateCtx::SetParameter

Upstream requirements: RS_CRYPTO_02102, RS_CRYPTO_02107, RS_CRYPTO_02108, RS_CRYPTO_02111

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/key_agreement_private_ctx.h"	
Scope:	`class ara::crypto::cryp::KeyAgreementPrivateCtx`	
Syntax:	`virtual ara::core::Result< void > SetParameter (ara::core::StringView parameterName, ReadOnlyMemRegion value) noexcept=0;`	
Parameters (in):	parameterName	an identifier of the parameter
	value	an input buffer containing the value of the parameter
Return value:	ara::core::Result< void >	an empty result on success or an error
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kUnsupported	rollback_semantics
		if the context does not support the parameter
	ara::crypto::CryptoErrc::kInvalidArgument	rollback_semantics
		if the provided parameter value is invalid
Description:	Set a parameter of the key agreement identified by parameterName.	

⌋

8.14.1.2.1.5 SetParameter

[SWS_CRYPT_50052] Definition of API function ara::crypto::cryp::KeyAgreementPrivateCtx::SetParameter

Upstream requirements: RS_CRYPTO_02102, RS_CRYPTO_02107, RS_CRYPTO_02108, RS_CRYPTO_02111

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/key_agreement_private_ctx.h"	
Scope:	`class ara::crypto::cryp::KeyAgreementPrivateCtx`	
Syntax:	`virtual ara::core::Result< void > SetParameter (ara::core::StringView parameterName, std::uint64_t value) noexcept=0;`	
Parameters (in):	parameterName	an identifier of the parameter
	value	an unsigned integer containing the value of the parameter
Return value:	ara::core::Result< void >	an empty result on success or an error
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kUnsupported	rollback_semantics
		if the context does not support the parameter

▽

△

	ara::crypto::CryptoErrc::kInvalidArgument	rollback_semantics
		if the provided parameter value is invalid
Description:	Set a parameter of the key agreement identified by parameterName.	

⌋

8.15 Header: ara/crypto/cryp/key_decapsulator_private_ctx.h

8.15.1 Class: KeyDecapsulatorPrivateCtx

[SWS_CRYPT_21400] Definition of API class ara::crypto::cryp::KeyDecapsulatorPrivateCtx

Upstream requirements: RS_CRYPTO_02104, RS_CRYPTO_02209

⌈

Kind:	class
Header file:	#include "ara/crypto/cryp/key_decapsulator_private_ctx.h"
Forwarding header file:	#include "ara/crypto/crypto_fwd.h"
Scope:	namespace ara::crypto::cryp
Symbol:	KeyDecapsulatorPrivateCtx
Base class:	CryptoContext
Syntax:	`class KeyDecapsulatorPrivateCtx : public CryptoContext {...};`
Description:	Asymmetric Key Encapsulation Mechanism (KEM) Private key Context interface.

⌋

8.15.1.1 Public Member Types

8.15.1.1.1 Type Alias: Sptr

[SWS_CRYPT_21401] Definition of API type ara::crypto::cryp::KeyDecapsulatorPrivateCtx::Sptr

Upstream requirements: RS_CRYPTO_02104

⌈

Kind:	type alias
Header file:	#include "ara/crypto/cryp/key_decapsulator_private_ctx.h"
Scope:	`class ara::crypto::cryp::KeyDecapsulatorPrivateCtx`
Symbol:	Sptr
Syntax:	`using Sptr = std::shared_ptr<KeyDecapsulatorPrivateCtx>;`
Description:	Smart pointer of the interface.

⌋

8.15.1.2 Public Member Functions

8.15.1.2.1 Member Functions

8.15.1.2.1.1 Decapsulate

[SWS_CRYPT_21412] Definition of API function ara::crypto::cryp::KeyDecapsulatorPrivateCtx::Decapsulate

Upstream requirements: RS_CRYPTO_02102, RS_CRYPTO_02108, RS_CRYPTO_02115

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/key_decapsulator_private_ctx.h"	
Scope:	`class ara::crypto::cryp::KeyDecapsulatorPrivateCtx`	
Syntax:	`virtual ara::core::Result< SymmetricKey::Sptr > Decapsulate (ReadOnly MemRegion encapsulatedKey, AlgId algId, AllowedUsageFlags allowed Usage) const noexcept=0;`	
Parameters (in):	encapsulatedKey	an input buffer containing the encapsulated key
	algId	the identifier of the target symmetric crypto algorithm
	allowedUsage	the flags that define a list of allowed transformations' types in which the target key can be used
Return value:	ara::core::Result< SymmetricKey::Sptr >	a result containing a the decapsulated symmetric key object or an error
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	CryptoErrc::kUninitializedContext	rollback_semantics
		if the context was not initialized by a private key value
	CryptoErrc::kInvalidArgument	rollback_semantics
		if the format of encapsulatedKey is invalid
	ara::crypto::CryptoErrc::kUnknownIdentifier	rollback_semantics
		if `algId` has an unsupported value
	ara::crypto::CryptoErrc::kIncompatibleArguments	rollback_semantics
		if the `allowedUsage` is incompatible with the `algId` or the key type
	ara::crypto::CryptoErrc::kIncompatibleArguments	rollback_semantics
		if the `algId` is incompatible to the encapsulatedKey
	ara::crypto::CryptoErrc::kBusyResource	rollback_semantics
		a required resource is busy
	ara::crypto::CryptoErrc::kEmptyKeySlot	rollback_semantics
		if a KeySlot was configured that is currently is empty
	ara::crypto::CryptoErrc::kUsageViolation	rollback_semantics
		if the allowed usage of the key does not include kAllowKeyUnwrapping
Description:	Decapsulate the keying data into a new SymmetricKey.	

⌋

8.15.1.2.1.2 SetKey

[SWS_CRYPT_21413] Definition of API function ara::crypto::cryp::KeyDecapsulatorPrivateCtx::SetKey

Upstream requirements: RS_CRYPTO_02001, RS_CRYPTO_02003

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/key_decapsulator_private_ctx.h"	
Scope:	`class ara::crypto::cryp::KeyDecapsulatorPrivateCtx`	
Syntax:	`virtual ara::core::Result< void > SetKey (PrivateKey::Sptr key) noexcept=0;`	
Parameters (in):	key	the key encapsulation key
Return value:	ara::core::Result< void >	An empty result or an error
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kIncompatibleObject	rollback_semantics
		if the provided key object is incompatible with this context
	ara::crypto::CryptoErrc::kUsageViolation	rollback_semantics
		if the allowed usage of the key does not include kAllowKeyUnwrapping
	ara::crypto::CryptoErrc::kUnsupported	rollback_semantics
		if the context only supports setting a KeySlot
Description:	Configure the context to use the provided key as decapsulation key.	

⌋

8.15.1.2.1.3 SetKey

[SWS_CRYPT_50083] Definition of API function ara::crypto::cryp::KeyDecapsulatorPrivateCtx::SetKey

Upstream requirements: RS_CRYPTO_02001, RS_CRYPTO_02003

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/key_decapsulator_private_ctx.h"	
Scope:	`class ara::crypto::cryp::KeyDecapsulatorPrivateCtx`	
Syntax:	`virtual ara::core::Result< void > SetKey (KeySlot::Sptr keySlot) noexcept=0;`	
Parameters (in):	keySlot	a keyslot containing a private key
Return value:	ara::core::Result< void >	An empty result or an error
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kIncompatibleObject	rollback_semantics
		if the provided keySlot is incompatible with this context

▽

△

	ara::crypto::CryptoErrc::kUnsupported	rollback_semantics if the context only supports setting a PrivateKey
Description:	Configure the context to use the key stored in the provided KeySlot at the time the key decapsulation is performed as encapsulation key.	

⌋

8.16 Header: ara/crypto/cryp/key_derivation_function_ctx.h

8.16.1 Class: KeyDerivationFunctionCtx

[SWS_CRYPT_21500] Definition of API class ara::crypto::cryp::KeyDerivationFunctionCtx

Upstream requirements: RS_CRYPTO_02103

⌈

Kind:	class
Header file:	#include "ara/crypto/cryp/key_derivation_function_ctx.h"
Forwarding header file:	#include "ara/crypto/crypto_fwd.h"
Scope:	namespace ara::crypto::cryp
Symbol:	KeyDerivationFunctionCtx
Base class:	CryptoContext
Syntax:	`class KeyDerivationFunctionCtx : public CryptoContext {...};`
Description:	Key Derivation Function interface.

⌋

8.16.1.1 Public Member Types

8.16.1.1.1 Type Alias: Sptr

[SWS_CRYPT_21501] Definition of API type ara::crypto::cryp::KeyDerivationFunctionCtx::Sptr

Upstream requirements: RS_CRYPTO_02103

⌈

Kind:	type alias
Header file:	#include "ara/crypto/cryp/key_derivation_function_ctx.h"
Scope:	`class ara::crypto::cryp::KeyDerivationFunctionCtx`
Symbol:	Sptr
Syntax:	`using Sptr = std::shared_ptr<KeyDerivationFunctionCtx>;`
Description:	Smart pointer of the interface.

⌋

8.16.1.2 Public Member Functions

8.16.1.2.1 Member Functions

8.16.1.2.1.1 DeriveKey

[SWS_CRYPT_21515] Definition of API function ara::crypto::cryp::KeyDerivationFunctionCtx::DeriveKey

Upstream requirements: RS_CRYPTO_02102, RS_CRYPTO_02107, RS_CRYPTO_02108, RS_CRYPTO_02111, RS_CRYPTO_02115

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/key_derivation_function_ctx.h"	
Scope:	`class ara::crypto::cryp::KeyDerivationFunctionCtx`	
Syntax:	`virtual ara::core::Result< SymmetricKey::Sptr > DeriveKey (AlgId targetAlgId, AllowedUsageFlags allowedUsage) const noexcept=0;`	
Parameters (in):	targetAlgId	the identifier of the target symmetric crypto algorithm
	allowedUsage	the flags that define a list of allowed usages in which the target key can be used.
Return value:	ara::core::Result< SymmetricKey::Sptr >	result containing a shared pointer to the created symmetric key object or an error
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kUnknownIdentifier	rollback_semantics
		if `targetAlgId` has an unsupported value
	ara::crypto::CryptoErrc::kIncompatibleArguments	rollback_semantics
		if the `allowedUsage` of the derived key is incompatible with the targetAlgId.
	ara::crypto::CryptoErrc::kUninitializedContext	rollback_semantics
		if the context has not been sufficiently initialized
	ara::crypto::CryptoErrc::kBusyResource	rollback_semantics
		a required resource is busy
	ara::crypto::CryptoErrc::kEmptyKeySlot	rollback_semantics
		if a KeySlot was configured that is currently is empty
	ara::crypto::CryptoErrc::kUsageViolation	rollback_semantics
		if the allowed usage of the source key does not include kAllowKeyDerivation
	ara::crypto::CryptoErrc::kUsageViolation	rollback_semantics
		if the allowedUsage of the target key includes kAllowPlaintextExport but the allowed usage of the source key does not include kAllowDerivedPlaintextExport
Description:	Derive a symmetric key from the provided key material and provided context configuration, e.g., salt, context, IV, iteration count.	

⌋

8.16.1.2.1.2 SetKey

[SWS_CRYPT_50010] Definition of API function ara::crypto::cryp::KeyDerivationFunctionCtx::SetKey

Upstream requirements: RS_CRYPTO_02001, RS_CRYPTO_02003

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/key_derivation_function_ctx.h"	
Scope:	`class ara::crypto::cryp::KeyDerivationFunctionCtx`	
Syntax:	`virtual ara::core::Result< void > SetKey (KeySlot::Sptr keySlot) noexcept=0;`	
Parameters (in):	keySlot	a keyslot containing a symmetric key
Return value:	ara::core::Result< void >	An empty result or an error
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kIncompatibleObject	rollback_semantics
		if the provided keySlot is incompatible with this context
	ara::crypto::CryptoErrc::kUnsupported	rollback_semantics
		if the context only supports setting a SymmetricKey
Description:	Configure the context to use the key stored in the provided KeySlot at the time the key derivation is performed.	

⌋

8.16.1.2.1.3 SetKey

[SWS_CRYPT_21525] Definition of API function ara::crypto::cryp::KeyDerivationFunctionCtx::SetKey

Upstream requirements: RS_CRYPTO_02001, RS_CRYPTO_02003

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/key_derivation_function_ctx.h"	
Scope:	`class ara::crypto::cryp::KeyDerivationFunctionCtx`	
Syntax:	`virtual ara::core::Result< void > SetKey (SymmetricKey::Sptr key) noexcept=0;`	
Parameters (in):	key	the source key-material
Return value:	ara::core::Result< void >	An empty result or an error
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kIncompatibleObject	rollback_semantics
		if the provided key object is incompatible with this symmetric key context
	ara::crypto::CryptoErrc::kUsageViolation	rollback_semantics

▽

△

		if the allowed usage of the key does not include kAllowKeyDerivation
	ara::crypto::CryptoErrc::kUnsupported	rollback_semantics
		if the context only supports setting a KeySlot
Description:	Configure the context to use the provided key.	

⌋

8.16.1.2.1.4 SetParameter

[SWS_CRYPT_50008] Definition of API function ara::crypto::cryp::KeyDerivationFunctionCtx::SetParameter

Upstream requirements: RS_CRYPTO_02102, RS_CRYPTO_02107, RS_CRYPTO_02108, RS_CRYPTO_02111

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/key_derivation_function_ctx.h"	
Scope:	`class ara::crypto::cryp::KeyDerivationFunctionCtx`	
Syntax:	`virtual ara::core::Result< void > SetParameter (ara::core::StringView parameterName, std::uint64_t value) noexcept=0;`	
Parameters (in):	parameterName	an identifier of the parameter
	value	an unsigned integer containing the value of the parameter
Return value:	ara::core::Result< void >	an empty result on success or an error
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kUnsupported	rollback_semantics
		if the context does not support the parameter
	ara::crypto::CryptoErrc::kInvalidArgument	rollback_semantics
		if the provided parameter value is invalid
Description:	Set a parameter of the KDF identified by parameterName.	

⌋

8.16.1.2.1.5 SetParameter

[SWS_CRYPT_50007] Definition of API function ara::crypto::cryp::KeyDerivationFunctionCtx::SetParameter

Upstream requirements: RS_CRYPTO_02102, RS_CRYPTO_02107, RS_CRYPTO_02108, RS_CRYPTO_02111

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/key_derivation_function_ctx.h"	
Scope:	`class ara::crypto::cryp::KeyDerivationFunctionCtx`	
Syntax:	`virtual ara::core::Result< void > SetParameter (ara::core::StringView parameterName, ReadOnlyMemRegion value) noexcept=0;`	
Parameters (in):	parameterName	an identifier of the parameter
	value	an input buffer containing the value of the parameter
Return value:	ara::core::Result< void >	an empty result on success or an error
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kUnsupported	rollback_semantics
		if the context does not support the parameter
	ara::crypto::CryptoErrc::kInvalidArgument	rollback_semantics
		if the provided parameter value is invalid
Description:	Set a parameter of the KDF identified by parameterName.	

⌋

8.16.1.2.1.6 SetParameter

[SWS_CRYPT_50009] Definition of API function ara::crypto::cryp::KeyDerivationFunctionCtx::SetParameter

Upstream requirements: RS_CRYPTO_02102, RS_CRYPTO_02107, RS_CRYPTO_02108, RS_CRYPTO_02111

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/key_derivation_function_ctx.h"	
Scope:	`class ara::crypto::cryp::KeyDerivationFunctionCtx`	
Syntax:	`virtual ara::core::Result< void > SetParameter (ara::core::StringView parameterName, SymmetricKey::Sptr value) noexcept=0;`	
Parameters (in):	parameterName	an identifier of the parameter
	value	SymmetricKey being used as the input parameter
Return value:	ara::core::Result< void >	an empty result on success or an error
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kUnsupported	rollback_semantics
		if the context does not support the parameter

▽

△

	ara::crypto::CryptoErrc::kIncompatibleObject	rollback_semantics
		if the provided key is incompatible with this context
	ara::crypto::CryptoErrc::kUsageViolation	rollback_semantics
		if the allowed usage of the key does not include kAllowKeyDerivation
Description:	Set a parameter of the KDF identified by parameterName.	

⌋

8.17 Header: ara/crypto/cryp/key_encapsulator_public_ctx.h

8.17.1 Class: KeyEncapsulatorPublicCtx

[SWS_CRYPT_21800] Definition of API class ara::crypto::cryp::KeyEncapsulatorPublicCtx

Upstream requirements: RS_CRYPTO_02104, RS_CRYPTO_02209

⌈

Kind:	class
Header file:	#include "ara/crypto/cryp/key_encapsulator_public_ctx.h"
Forwarding header file:	#include "ara/crypto/crypto_fwd.h"
Scope:	namespace ara::crypto::cryp
Symbol:	KeyEncapsulatorPublicCtx
Base class:	CryptoContext
Syntax:	`class KeyEncapsulatorPublicCtx : public CryptoContext {...};`
Description:	Asymmetric Key Encapsulation Mechanism (KEM) Public key Context interface.

⌋

8.17.1.1 Public Member Types

8.17.1.1.1 Type Alias: Sptr

[SWS_CRYPT_21801] Definition of API type ara::crypto::cryp::KeyEncapsulatorPublicCtx::Sptr

Upstream requirements: RS_CRYPTO_02209

⌈

Kind:	type alias
Header file:	#include "ara/crypto/cryp/key_encapsulator_public_ctx.h"
Scope:	`class ara::crypto::cryp::KeyEncapsulatorPublicCtx`
Symbol:	Sptr

▽

△

Syntax:	`using Sptr = std::shared_ptr<KeyEncapsulatorPublicCtx>;`
Description:	Smart pointer of the interface.

⌋

8.17.1.2 Public Member Functions

8.17.1.2.1 Member Functions

8.17.1.2.1.1 Encapsulate

[SWS_CRYPT_21813] Definition of API function ara::crypto::cryp::KeyEncapsulatorPublicCtx::Encapsulate

Upstream requirements: RS_CRYPTO_02102, RS_CRYPTO_02108, RS_CRYPTO_02115

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/key_encapsulator_public_ctx.h"	
Scope:	`class ara::crypto::cryp::KeyEncapsulatorPublicCtx`	
Syntax:	`virtual ara::core::Result< std::size_t > Encapsulate (SymmetricKey::Sptr key, ReadWriteMemRegion encapsulatedKey) const noexcept=0;`	
Parameters (in):	key	the symmetric key to be encapsulated
Parameters (out):	encapsulatedKey	an output buffer the encapsulated key is written to
Return value:	ara::core::Result< std::size_t >	a result containing the size of the encapsulated key or an error
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	CryptoErrc::kUninitializedContext	rollback_semantics
		if the context was not initialized by a public key value
	ara::crypto::CryptoErrc::kUsageViolation	rollback_semantics
		if the allowed usage of the encapsulation key does not include kAllowKeyWrapping or if the allowed usage of the encapsulated key does not include kAllowSecuredExport
	ara::crypto::CryptoErrc::kIncompatibleObject	rollback_semantics
		if the provided encapsulated key object is incompatible with this context
	CryptoErrc::kInvalidInputSize	rollback_semantics
		if this context does not support the size of the encapsulated key
	CryptoErrc::kUnsupported	rollback_semantics
		if this context does not support setting the encapsulated key, because it generates the key during encapsulation
	ara::crypto::CryptoErrc::kInsufficientCapacity	rollback_semantics
		if the output buffer is too small
	ara::crypto::CryptoErrc::kBusyResource	rollback_semantics
		a required resource is busy

▽

△

	ara::crypto::CryptoErrc::kEmptyKeySlot	rollback_semantics
		if a KeySlot was configured that is currently is empty
	ara::crypto::CryptoErrc::kUsageViolation	rollback_semantics
		if the allowed usage of the key does not include kAllowKeyWrapping
Description:	Encapsulate the provided symmetric key.	

⌋

8.17.1.2.1.2 Encapsulate

[SWS_CRYPT_50011] Definition of API function ara::crypto::cryp::KeyEncapsulatorPublicCtx::Encapsulate

Upstream requirements: RS_CRYPTO_02102, RS_CRYPTO_02108, RS_CRYPTO_02115

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/key_encapsulator_public_ctx.h"	
Scope:	`class ara::crypto::cryp::KeyEncapsulatorPublicCtx`	
Syntax:	`virtual ara::core::Result< std::pair< std::size_t, SymmetricKey::Sptr > > Encapsulate (AlgId algId, AllowedUsageFlags allowedUsage, ReadWriteMemRegion encapsulatedKey) const noexcept=0;`	
Parameters (in):	algId	the identifier of the target symmetric crypto algorithm
	allowedUsage	the flags that define a list of allowed transformations' types in which the target key can be used
Parameters (out):	encapsulatedKey	an output buffer the encapsulated key is written to
Return value:	ara::core::Result< std::pair< std::size_t, SymmetricKey::Sptr > >	a result containing a pair of the size of the encapsulated key and the generated symmetric key or an error
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	CryptoErrc::kUninitializedContext	rollback_semantics
		if the context was not initialized by a public key value
	ara::crypto::CryptoErrc::kUsageViolation	rollback_semantics
		if the allowed usage of the encapsulation key does not include k AllowKeyWrapping or if the allowed usage of the encapsulated key does not include kAllowSecuredExport
	CryptoErrc::kInvalidInputSize	rollback_semantics
		if this context does not support the size of the encapsulated key
	CryptoErrc::kUnsupported	rollback_semantics
		if this context does not support generating the encapsulated key during encapsulation
	ara::crypto::CryptoErrc::kUnknownIdentifier	rollback_semantics
		if `algId` has an unsupported value
	ara::crypto::CryptoErrc::kInsufficientCapacity	rollback_semantics
		if the output buffer is too small

▽

△

	ara::crypto::CryptoErrc::kBusyResource	rollback_semantics
		a required resource is busy
	ara::crypto::CryptoErrc::kEmptyKeySlot	rollback_semantics
		if a KeySlot was configured via SetKey that is currently is empty
	ara::crypto::CryptoErrc::kUsageViolation	rollback_semantics
		if the allowed usage of the key does not include kAllowKeyWrapping
Description:	Generate new a symmetric key and encapsulate it. Optionally store the generated key.	

⌋

8.17.1.2.1.3 GetRequiredOutputBufferSize

[SWS_CRYPT_50013] Definition of API function ara::crypto::cryp::KeyEncapsulatorPublicCtx::GetRequiredOutputBufferSize

Upstream requirements: RS_CRYPTO_02302

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/key_encapsulator_public_ctx.h"	
Scope:	`class ara::crypto::cryp::KeyEncapsulatorPublicCtx`	
Syntax:	`virtual ara::core::Result< std::size_t > GetRequiredOutputBufferSize (AlgId algId) const noexcept;`	
Parameters (in):	algId	the algId of the key to encapsulate
Return value:	ara::core::Result< std::size_t >	a result containing the required capacity of the output buffer in bytes or an error
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kUninitializedContext	rollback_semantics
		if the context was not initialized by deploying a key
Description:	Calculate the maximum required size of the output buffer depending on the input size in the current state of the context.	

⌋

8.17.1.2.1.4 SetKey

[SWS_CRYPT_21815] Definition of API function ara::crypto::cryp::KeyEncapsulatorPublicCtx::SetKey

Upstream requirements: RS_CRYPTO_02001, RS_CRYPTO_02003

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/key_encapsulator_public_ctx.h"	
Scope:	`class ara::crypto::cryp::KeyEncapsulatorPublicCtx`	
Syntax:	`virtual ara::core::Result< void > SetKey (PublicKey::Sptr key) noexcept=0;`	
Parameters (in):	key	the key encapsulation key
Return value:	ara::core::Result< void >	An empty result or an error
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kIncompatibleObject	rollback_semantics
		if the provided key object is incompatible with this context
	ara::crypto::CryptoErrc::kUsageViolation	rollback_semantics
		if the allowed usage of the key does not include kAllowKeyWrapping
	ara::crypto::CryptoErrc::kUnsupported	rollback_semantics
		if the context only supports setting a KeySlot
Description:	Configure the context to use the provided key as encapsulation key.	

⌋

8.17.1.2.1.5 SetKey

[SWS_CRYPT_50012] Definition of API function ara::crypto::cryp::KeyEncapsulatorPublicCtx::SetKey

Upstream requirements: RS_CRYPTO_02001, RS_CRYPTO_02003

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/key_encapsulator_public_ctx.h"	
Scope:	`class ara::crypto::cryp::KeyEncapsulatorPublicCtx`	
Syntax:	`virtual ara::core::Result< void > SetKey (KeySlot::Sptr keySlot) noexcept=0;`	
Parameters (in):	keySlot	a keyslot containing a public key
Return value:	ara::core::Result< void >	An empty result or an error
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kIncompatibleObject	rollback_semantics
		if the provided keySlot is incompatible with this context

▽

△

	ara::crypto::CryptoErrc::kUnsupported	rollback_semantics
		if the context only supports setting a PublicKey
Description:	Configure the context to use the key stored in the provided KeySlot at the time the key encapsulation is performed.	

⌋

8.18 Header: ara/crypto/cryp/keyslot.h

8.18.1 Class: KeySlot

[SWS_CRYPT_50145] Definition of API class ara::crypto::cryp::KeySlot

Upstream requirements: RS_CRYPTO_02405

⌈

Kind:	class
Port Interfaces:	`CryptoKeySlotClientInterface`
Header file:	#include "ara/crypto/cryp/keyslot.h"
Forwarding header file:	#include "ara/crypto/crypto_fwd.h"
Scope:	namespace ara::crypto::cryp
Symbol:	KeySlot
Syntax:	`class KeySlot {...};`
Description:	KeySlot interface representing a storage location for a key in a CryptoProvider.

⌋

8.18.1.1 Public Member Types

8.18.1.1.1 Type Alias: Sptr

[SWS_CRYPT_50151] Definition of API type ara::crypto::cryp::KeySlot::Sptr

Upstream requirements: RS_CRYPTO_02405

⌈

Kind:	type alias
Header file:	#include "ara/crypto/cryp/keyslot.h"
Scope:	`class ara::crypto::cryp::KeySlot`
Symbol:	Sptr
Syntax:	`using Sptr = std::shared_ptr<KeySlot>;`
Description:	Smart pointer of the interface.

⌋

8.18.1.2 Public Member Functions

8.18.1.2.1 Special Member Functions

8.18.1.2.1.1 Copy Constructor

[SWS_CRYPT_50149] Definition of API function ara::crypto::cryp::KeySlot::KeySlot

Upstream requirements: RS_CRYPTO_02004

⌈

Kind:	function
Header file:	#include "ara/crypto/cryp/keyslot.h"
Scope:	`class ara::crypto::cryp::KeySlot`
Syntax:	`KeySlot (const KeySlot &)=delete;`
Description:	Copy-Constructor.

⌋

8.18.1.2.1.2 Move Constructor

[SWS_CRYPT_50150] Definition of API function ara::crypto::cryp::KeySlot::KeySlot

Upstream requirements: RS_CRYPTO_02004

⌈

Kind:	function
Header file:	#include "ara/crypto/cryp/keyslot.h"
Scope:	`class ara::crypto::cryp::KeySlot`
Syntax:	`KeySlot (KeySlot &&)=delete;`
Description:	Move-Constructor.

⌋

8.18.1.2.1.3 Move Assignment Operator

[SWS_CRYPT_50148] Definition of API function ara::crypto::cryp::KeySlot::operator=

Upstream requirements: RS_CRYPTO_02004

⌈

Kind:	function
Header file:	#include "ara/crypto/cryp/keyslot.h"
Scope:	`class ara::crypto::cryp::KeySlot`
Syntax:	`KeySlot & operator= (KeySlot &&other)=delete;`
Description:	Move-assign another KeySlot to this instance.

⌋

8.18.1.2.1.4 Copy Assignment Operator

[SWS_CRYPT_50147] Definition of API function ara::crypto::cryp::KeySlot::operator=

Upstream requirements: RS_CRYPTO_02004

⌈

Kind:	function
Header file:	#include "ara/crypto/cryp/keyslot.h"
Scope:	`class ara::crypto::cryp::KeySlot`
Syntax:	`KeySlot & operator= (const KeySlot &other)=delete;`
Description:	Copy-assign another KeySlot to this instance.

⌋

8.18.1.2.1.5 Destructor

[SWS_CRYPT_50146] Definition of API function ara::crypto::cryp::KeySlot::~KeySlot

Upstream requirements: RS_CRYPTO_02405

⌈

Kind:	function
Header file:	#include "ara/crypto/cryp/keyslot.h"
Scope:	`class ara::crypto::cryp::KeySlot`
Syntax:	`virtual ~KeySlot () noexcept=default;`
Exception Safety:	exception safe
Description:	Destructor.

⌋

8.18.1.2.2 Member Functions

8.18.1.2.2.1 GetAlgId

[SWS_CRYPT_50055] Definition of API function ara::crypto::cryp::KeySlot::GetAlgId

Upstream requirements: RS_CRYPTO_02009

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/keyslot.h"	
Scope:	`class ara::crypto::cryp::KeySlot`	
Syntax:	`virtual ara::core::Result< ara::crypto::AlgIdString > GetAlgId () const noexcept=0;`	
Return value:	ara::core::Result< ara::crypto::AlgIdString >	A result containing the AlgId of the current key slot content
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kEmptyKeySlot	rollback_semantics
		if the KeySlot does not contain a key.
Description:	The interface shall return the AlgId of the currently stored key.	

⌋

8.18.1.2.2.2 GetCryptoObjectType

[SWS_CRYPT_50054] Definition of API function ara::crypto::cryp::KeySlot::GetCryptoObjectType

Upstream requirements: RS_CRYPTO_02009

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/keyslot.h"	
Scope:	`class ara::crypto::cryp::KeySlot`	
Syntax:	`virtual ara::core::Result< CryptoObjectType > GetCryptoObjectType () const noexcept=0;`	
Return value:	ara::core::Result< CryptoObjectType >	A result containing the CryptoObjectType of the currently stored key.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kEmptyKeySlot	rollback_semantics
		if the KeySlot does not contain a key.
Description:	The interface shall return the CryptoObjectType of the currently stored key.	

⌋

8.18.1.2.2.3 GetCryptoProvider

[SWS_CRYPT_30403] Definition of API function ara::crypto::cryp::KeySlot::GetCryptoProvider

Upstream requirements: RS_CRYPTO_02401

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/keyslot.h"	
Scope:	`class ara::crypto::cryp::KeySlot`	
Syntax:	`virtual ara::core::Result< std::shared_ptr< cryp::CryptoProvider > > GetCryptoProvider () const noexcept=0;`	
Return value:	ara::core::Result< std::shared_ptr< cryp::CryptoProvider > >	a result containing a shared pointer to the CryptoProvider to be used with this KeySlot
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	This function does not specify any standardized errors.	
Description:	The interface shall return an instance of the CryptoProvider that is responsible for this KeySlot.	

⌋

8.18.1.2.2.4 GetPossibleAlgIds

[SWS_CRYPT_50056] Definition of API function ara::crypto::cryp::KeySlot::GetPossibleAlgIds

Upstream requirements: RS_CRYPTO_02009

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/keyslot.h"	
Scope:	`class ara::crypto::cryp::KeySlot`	
Syntax:	`virtual ara::core::Result< ara::core::Vector< ara::crypto::AlgIdString > > GetPossibleAlgIds () const noexcept=0;`	
Return value:	ara::core::Result< ara::core::Vector< ara::crypto::AlgIdString > >	A result containing the possible AlgIds of the key slot
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	This function does not specify any standardized errors.	
Description:	The interface shall return a vector of AlgIds supported by this KeySlot.	

⌋

8.18.1.2.2.5 RegisterUpdateObserver

[SWS_CRYPT_30404] Definition of API function ara::crypto::cryp::KeySlot::RegisterUpdateObserver

Upstream requirements: RS_CRYPTO_02401

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/keyslot.h"	
Scope:	`class ara::crypto::cryp::KeySlot`	
Syntax:	`virtual ara::core::Result< void > RegisterUpdateObserver (std::function< void(ara::core::StringView)> callback, ara::core::StringView keySlotIdentifier) const noexcept=0;`	
Parameters (in):	callback	the registered callback function to be called when the KeySlot content is modified
	keySlotIdentifier	an application-specific identified of the KeySlot which will be used as a parameter of the callback function
Return value:	ara::core::Result< void >	an empty result or an implementation specific error.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	This function does not specify any standardized errors.	
Description:	The interface shall register an application callback function to be called when the content of the KeySlot is updated (i.e., modified or cleared). If this function is called multiple times, only the most recently registered callback of an application will be called.	

⌋

8.18.1.2.2.6 UnregisterUpdateObserver

[SWS_CRYPT_30406] Definition of API function ara::crypto::cryp::KeySlot::UnregisterUpdateObserver

Upstream requirements: RS_CRYPTO_02401

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/keyslot.h"	
Scope:	`class ara::crypto::cryp::KeySlot`	
Syntax:	`virtual ara::core::Result< void > UnregisterUpdateObserver () const noexcept=0;`	
Return value:	ara::core::Result< void >	an empty result or an implementation specific error.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	This function does not specify any standardized errors.	
Description:	Unregisters any previously UpdateObserver callback that has been registered previously by the application.	

⌋

8.19 Header: ara/crypto/cryp/message_authn_code_ctx.h

8.19.1 Class: MessageAuthnCodeCtx

[SWS_CRYPT_22100] Definition of API class ara::crypto::cryp::MessageAuthnCodeCtx

Upstream requirements: RS_CRYPTO_02203

⌈

Kind:	class
Header file:	#include "ara/crypto/cryp/message_authn_code_ctx.h"
Forwarding header file:	#include "ara/crypto/crypto_fwd.h"
Scope:	namespace ara::crypto::cryp
Symbol:	MessageAuthnCodeCtx
Base class:	CryptoContext
Syntax:	`class MessageAuthnCodeCtx : public CryptoContext {...};`
Description:	Keyed Message Authentication Code Context interface definition.

⌋

8.19.1.1 Public Member Types

8.19.1.1.1 Type Alias: Sptr

[SWS_CRYPT_22101] Definition of API type ara::crypto::cryp::MessageAuthnCodeCtx::Sptr

Upstream requirements: RS_CRYPTO_02203

⌈

Kind:	type alias
Header file:	#include "ara/crypto/cryp/message_authn_code_ctx.h"
Scope:	`class ara::crypto::cryp::MessageAuthnCodeCtx`
Symbol:	Sptr
Syntax:	`using Sptr = std::shared_ptr<MessageAuthnCodeCtx>;`
Description:	Smart pointer of the interface.

⌋

8.19.1.2 Public Member Functions

8.19.1.2.1 Member Functions

8.19.1.2.1.1 Cancel

[SWS_CRYPT_50027] Definition of API function ara::crypto::cryp::MessageAuthnCodeCtx::Cancel

Upstream requirements: RS_CRYPTO_02302

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/message_authn_code_ctx.h"	
Scope:	`class ara::crypto::cryp::MessageAuthnCodeCtx`	
Syntax:	`virtual ara::core::Result< void > Cancel () noexcept=0;`	
Return value:	ara::core::Result< void >	an empty result if the ongoing streaming operation was cancelled successfully, or a result containing an error
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kProcessingNotStarted	rollback_semantics
		if data processing was not started by a call of the `Start()` method
Description:	Cancel the ongoing streaming operation that has been initiated by Start() and has not yet been finished.	

⌋

8.19.1.2.1.2 Compare

[SWS_CRYPT_50133] Definition of API function ara::crypto::cryp::MessageAuthnCodeCtx::Compare

Upstream requirements: RS_CRYPTO_02309

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/message_authn_code_ctx.h"	
Scope:	`class ara::crypto::cryp::MessageAuthnCodeCtx`	
Syntax:	`virtual ara::core::Result< bool > Compare (ReadOnlyMemRegion expected) const noexcept=0;`	
Parameters (in):	expected	the expected MAC digest
Return value:	ara::core::Result< bool >	`true` if the expected MAC value equals the calculated MAC value, false otherwise.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kProcessingNotFinished	rollback_semantics
		if the digest calculation was not finished by a call of the `Finish()` method

▽

△

	ara::crypto::CryptoErrc::kInvalidInputSize	rollback_semantics
		if the size of the expected MAC digest is not supported
Description:	Compare the size and content of the calculated MAC digest against an expected value.	

⌋

8.19.1.2.1.3 CompareSubspan

[SWS_CRYPT_50028] Definition of API function ara::crypto::cryp::MessageAuthnCodeCtx::CompareSubspan

Upstream requirements: RS_CRYPTO_02309

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/message_authn_code_ctx.h"	
Scope:	`class ara::crypto::cryp::MessageAuthnCodeCtx`	
Syntax:	`virtual ara::core::Result< bool > CompareSubspan (ReadOnlyMemRegion expected) const noexcept=0;`	
Parameters (in):	expected	the expected MAC digest
Return value:	ara::core::Result< bool >	`true` if the most-significant bits of the calculated MAC equal the expected byte sequence, false otherwise.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kProcessingNotFinished	rollback_semantics
		if the digest calculation was not finished by a call of the `Finish()` method
	ara::crypto::CryptoErrc::kInvalidInputSize	rollback_semantics
		if the size of the expected MAC digest is not supported
Description:	Compare the most significant bits of the calculated MAC digest against an expected value.	

⌋

8.19.1.2.1.4 Finish

[SWS_CRYPT_22115] Definition of API function ara::crypto::cryp::MessageAuthnCodeCtx::Finish

Upstream requirements: RS_CRYPTO_02302, RS_CRYPTO_02203

⌈

Kind:	function
Header file:	#include "ara/crypto/cryp/message_authn_code_ctx.h"
Scope:	`class ara::crypto::cryp::MessageAuthnCodeCtx`
Syntax:	`virtual ara::core::Result< void > Finish () noexcept=0;`

▽

△

Return value:	ara::core::Result< void >	An empty result or an error
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kProcessingNotStarted	rollback_semantics
		if the MAC calculation has not been started successfully.
Description:	Finish the MAC calculation.	

⌋

8.19.1.2.1.5 Generate

[SWS_CRYPT_50029] Definition of API function ara::crypto::cryp::MessageAuthnCodeCtx::Generate

Upstream requirements: RS_CRYPTO_02309

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/message_authn_code_ctx.h"	
Scope:	`class ara::crypto::cryp::MessageAuthnCodeCtx`	
Syntax:	`virtual ara::core::Result< std::size_t > Generate (ReadOnlyMemRegion data, ReadWriteMemRegion digest, ara::core::Optional< ReadOnlyMemRegion > iv) const noexcept=0;`	
Parameters (in):	data	the input data for the MAC
	iv	an optional initialization vector
Parameters (out):	digest	an output data buffer that the MAC digest will be written into
Return value:	ara::core::Result< std::size_t >	A result containing the size in bytes of the MAC digest or an error.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kUninitializedContext	rollback_semantics
		if the context was not initialized by deploying a key
	ara::crypto::CryptoErrc::kUnsupported	rollback_semantics
		if single-call processing is not supported, or if the context requires setting an IV but no IV is provided, or if the context does not support setting an IV but an IV is provided
	ara::crypto::CryptoErrc::kInvalidInputSize	rollback_semantics
		if the size of provided data or IV is not supported
	ara::crypto::CryptoErrc::kInsufficientCapacity	rollback_semantics
		if the output buffer is too small
	ara::crypto::CryptoErrc::kBusyResource	rollback_semantics
		a required resource is busy
	ara::crypto::CryptoErrc::kEmptyKeySlot	rollback_semantics
		if a KeySlot was configured that is currently is empty

▽

△

	ara::crypto::CryptoErrc::kUsageViolation	rollback_semantics
		if the allowed usage of the key does not include kAllowMac
Description:	Calculate a MAC digest and write it to the provided output buffer.	

⌋

8.19.1.2.1.6 GenerateSubspan

[SWS_CRYPT_50030] Definition of API function ara::crypto::cryp::MessageAuthnCodeCtx::GenerateSubspan

Upstream requirements: RS_CRYPTO_02309

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/message_authn_code_ctx.h"	
Scope:	`class ara::crypto::cryp::MessageAuthnCodeCtx`	
Syntax:	`virtual ara::core::Result< std::size_t > GenerateSubspan (ReadOnlyMemRegion data, ReadWriteMemRegion truncatedDigest, ara::core::Optional< ReadOnlyMemRegion > iv) const noexcept=0;`	
Parameters (in):	data	the input data for the MAC
	iv	an optional initialization vector
Parameters (out):	truncatedDigest	an output data buffer that the beginning of the MAC digest will be written into
Return value:	ara::core::Result< std::size_t >	A result containing the size in bytes of the MAC digest or an error.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kUninitializedContext	rollback_semantics
		if the context was not initialized by deploying a key
	ara::crypto::CryptoErrc::kUnsupported	rollback_semantics
		if single-call processing is not supported, or if the context requires setting an IV but no IV is provided, or if the context does not support setting an IV but an IV is provided
	ara::crypto::CryptoErrc::kInvalidInputSize	rollback_semantics
		if the size of provided data or IV is not supported
	ara::crypto::CryptoErrc::kBusyResource	rollback_semantics
		a required resource is busy
	ara::crypto::CryptoErrc::kEmptyKeySlot	rollback_semantics
		if a KeySlot was configured that is currently is empty
	ara::crypto::CryptoErrc::kUsageViolation	rollback_semantics
		if the allowed usage of the key does not include kAllowMac
Description:	Calculate a MAC digest and fill the provided output buffer with the beginning (MSB) of the calculated digest.	

⌋

8.19.1.2.1.7 GetDigest

[SWS_CRYPT_22116] Definition of API function ara::crypto::cryp::MessageAuthnCodeCtx::GetDigest

Upstream requirements: RS_CRYPTO_02203

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/message_authn_code_ctx.h"	
Scope:	`class ara::crypto::cryp::MessageAuthnCodeCtx`	
Syntax:	`virtual ara::core::Result< std::size_t > GetDigest (ReadWriteMemRegion digest) const noexcept=0;`	
Parameters (out):	digest	an output data buffer that the MAC digest will be written into
Return value:	ara::core::Result< std::size_t >	A result containing the size in bytes of the MAC digest or an error.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kProcessingNotFinished	rollback_semantics
		if the MAC calculation was not finished by a call of the `Finish()` method
	ara::crypto::CryptoErrc::kInsufficientCapacity	rollback_semantics
		if the output buffer is too small
Description:	Write the calculated MAC digest into the provided output buffer.	

⌋

8.19.1.2.1.8 GetDigestSize

[SWS_CRYPT_50134] Definition of API function ara::crypto::cryp::MessageAuthnCodeCtx::GetDigestSize

Upstream requirements: RS_CRYPTO_02309

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/message_authn_code_ctx.h"	
Scope:	`class ara::crypto::cryp::MessageAuthnCodeCtx`	
Syntax:	`virtual ara::core::Result< std::size_t > GetDigestSize () const noexcept=0;`	
Return value:	ara::core::Result< std::size_t >	size of the hash digest in bytes
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	This function does not specify any standardized errors.	
Description:	Get the output digest size.	

⌋

8.19.1.2.1.9 GetMaximumChunkSize

[SWS_CRYPT_50033] Definition of API function ara::crypto::cryp::MessageAuthnCodeCtx::GetMaximumChunkSize

Upstream requirements: RS_CRYPTO_02309

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/message_authn_code_ctx.h"	
Scope:	`class ara::crypto::cryp::MessageAuthnCodeCtx`	
Syntax:	`virtual ara::core::Result< std::size_t > GetMaximumChunkSize () const noexcept=0;`	
Return value:	ara::core::Result< std::size_t >	the maximum buffer size in bytes
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	This function does not specify any standardized errors.	
Description:	Get the maximum input buffer size supported for processing in single calls to Update() or Finish() by this context.	

⌋

8.19.1.2.1.10 GetRecommendedIvSize

[SWS_CRYPT_50135] Definition of API function ara::crypto::cryp::MessageAuthnCodeCtx::GetRecommendedIvSize

Upstream requirements: RS_CRYPTO_02309

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/message_authn_code_ctx.h"	
Scope:	`class ara::crypto::cryp::MessageAuthnCodeCtx`	
Syntax:	`virtual ara::core::Result< std::size_t > GetRecommendedIvSize () const noexcept=0;`	
Return value:	ara::core::Result< std::size_t >	default expected size of IV in bytes
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	This function does not specify any standardized errors.	
Description:	Get default expected size of the Initialization Vector (IV) or nonce, e.g., 12 bytes for GMAC, 0 bytes for CMAC.	

⌋

8.19.1.2.1.11 SetKey

[SWS_CRYPT_50026] Definition of API function ara::crypto::cryp::MessageAuthnCodeCtx::SetKey

Upstream requirements: RS_CRYPTO_02001, RS_CRYPTO_02003

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/message_authn_code_ctx.h"	
Scope:	`class ara::crypto::cryp::MessageAuthnCodeCtx`	
Syntax:	`virtual ara::core::Result< void > SetKey (KeySlot::Sptr keySlot) noexcept=0;`	
Parameters (in):	keySlot	a keyslot containing a symmetric key
Return value:	ara::core::Result< void >	An empty result or an error
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kIncompatibleObject	rollback_semantics
		if the provided keySlot is incompatible with this symmetric key context
	ara::crypto::CryptoErrc::kUnsupported	rollback_semantics
		if the context only supports setting a SymmetricKey
Description:	Configure the context to use the key stored in the provided KeySlot at the time the MAC calculation is started.	

⌋

8.19.1.2.1.12 SetKey

[SWS_CRYPT_22118] Definition of API function ara::crypto::cryp::MessageAuthnCodeCtx::SetKey

Upstream requirements: RS_CRYPTO_02001, RS_CRYPTO_02003

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/message_authn_code_ctx.h"	
Scope:	`class ara::crypto::cryp::MessageAuthnCodeCtx`	
Syntax:	`virtual ara::core::Result< void > SetKey (SymmetricKey::Sptr key) noexcept=0;`	
Parameters (in):	key	symmetric key object used for MAC calculation
Return value:	ara::core::Result< void >	An empty result or an error
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kUnsupported	rollback_semantics
		if the context only supports setting a KeySlot
	ara::crypto::CryptoErrc::kIncompatibleObject	rollback_semantics
		if the provided key object is incompatible with this context

▽

△

	ara::crypto::CryptoErrc::kUsageViolation	rollback_semantics
		if the allowed usage of the key does not include kAllowMac
Description:	Configure the context to use the provided key.	

⌋

8.19.1.2.1.13 Start

[SWS_CRYPT_22110] Definition of API function ara::crypto::cryp::MessageAuthnCodeCtx::Start

Upstream requirements: RS_CRYPTO_02302

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/message_authn_code_ctx.h"	
Scope:	`class ara::crypto::cryp::MessageAuthnCodeCtx`	
Syntax:	`virtual ara::core::Result< void > Start (ReadOnlyMemRegion iv) noexcept=0;`	
Parameters (in):	iv	an Initialization Vector (IV) or "nonce" value
Return value:	ara::core::Result< void >	An empty result or an error
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kUninitializedContext	rollback_semantics
		if the context was not initialized by deploying a key
	ara::crypto::CryptoErrc::kInvalidInputSize	rollback_semantics
		if the size of provided IV is not supported
	ara::crypto::CryptoErrc::kUnsupported	rollback_semantics
		if the context does not support streaming or the algorithm does not support setting an IV
	ara::crypto::CryptoErrc::kBusyResource	rollback_semantics
		a required resource is busy
	ara::crypto::CryptoErrc::kEmptyKeySlot	rollback_semantics
		if a KeySlot was configured that is currently is empty
	ara::crypto::CryptoErrc::kUsageViolation	rollback_semantics
		if the allowed usage of the key does not include kAllowMac
Description:	Initialize the context for a new MAC calculation and set an IV (e.g., used for GMAC)	

⌋

8.19.1.2.1.14 Start

[SWS_CRYPT_22111] Definition of API function ara::crypto::cryp::MessageAuthnCodeCtx::Start

Upstream requirements: RS_CRYPTO_02302

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/message_authn_code_ctx.h"	
Scope:	`class ara::crypto::cryp::MessageAuthnCodeCtx`	
Syntax:	`virtual ara::core::Result< void > Start () noexcept=0;`	
Return value:	ara::core::Result< void >	An empty result or an error
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kUninitializedContext	rollback_semantics
		if the context was not initialized by deploying a key
	ara::crypto::CryptoErrc::kUnsupported	rollback_semantics
		if the context does not support streaming or the algorithm requires setting an IV
	ara::crypto::CryptoErrc::kBusyResource	rollback_semantics
		a required resource is busy
	ara::crypto::CryptoErrc::kEmptyKeySlot	rollback_semantics
		if a KeySlot was configured that is currently is empty
	ara::crypto::CryptoErrc::kUsageViolation	rollback_semantics
		if the allowed usage of the key does not include kAllowMac
Description:	Initialize the context for a new MAC calculation.	

⌋

8.19.1.2.1.15 Update

[SWS_CRYPT_22113] Definition of API function ara::crypto::cryp::MessageAuthnCodeCtx::Update

Upstream requirements: RS_CRYPTO_02302

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/message_authn_code_ctx.h"	
Scope:	`class ara::crypto::cryp::MessageAuthnCodeCtx`	
Syntax:	`virtual ara::core::Result< void > Update (ReadOnlyMemRegion data) noexcept=0;`	
Parameters (in):	data	a part of the input message that should be processed
Return value:	ara::core::Result< void >	An empty result or an error
Exception Safety:	exception safe	
Thread Safety:	thread-safe	

▽

△

Errors:	ara::crypto::CryptoErrc::kProcessingNotStarted	rollback_semantics
		if the digest calculation was not initiated by a call of the `Start()` method
	ara::crypto::CryptoErrc::kInvalidInputSize	rollback_semantics
		if the size of provided data is not supported
Description:	Update the MAC calculation context by a new part of the message.	

⌋

8.19.1.2.1.16 Verify

[SWS_CRYPT_50031] Definition of API function ara::crypto::cryp::MessageAuthnCodeCtx::Verify

Upstream requirements: RS_CRYPTO_02309

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/message_authn_code_ctx.h"	
Scope:	`class ara::crypto::cryp::MessageAuthnCodeCtx`	
Syntax:	`virtual ara::core::Result< bool > Verify (ReadOnlyMemRegion data, ReadOnlyMemRegion expected, ara::core::Optional< ReadOnlyMemRegion > iv) const noexcept=0;`	
Parameters (in):	data	the input data for the MAC
	expected	the expected MAC digest
	iv	an optional initialization vector
Return value:	ara::core::Result< bool >	`true` if the calculated MAC equals the expected byte sequence, false otherwise.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kUninitializedContext	rollback_semantics
		if the context was not initialized by deploying a key
	ara::crypto::CryptoErrc::kUnsupported	rollback_semantics
		if single-call operation is not supported, or if the context requires setting an IV but no IV is provided, or if the context does not support setting an IV but an IV is provided
	ara::crypto::CryptoErrc::kInvalidInputSize	rollback_semantics
		if the size of the provided data, expected digest or IV is not supported
	ara::crypto::CryptoErrc::kBusyResource	rollback_semantics
		a required resource is busy
	ara::crypto::CryptoErrc::kEmptyKeySlot	rollback_semantics
		if a KeySlot was configured that is currently is empty
	ara::crypto::CryptoErrc::kUsageViolation	rollback_semantics
		if the allowed usage of the key does not include kAllowMac
Description:	Calculate a MAC digest and compare the calculated MAC digest against an expected value.	

⌋

8.19.1.2.1.17 VerifySubspan

[SWS_CRYPT_50032] Definition of API function ara::crypto::cryp::MessageAuthnCodeCtx::VerifySubspan

Upstream requirements: RS_CRYPTO_02309

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/message_authn_code_ctx.h"	
Scope:	`class ara::crypto::cryp::MessageAuthnCodeCtx`	
Syntax:	`virtual ara::core::Result< bool > VerifySubspan (ReadOnlyMemRegion data, ReadOnlyMemRegion expected, ara::core::Optional< ReadOnlyMemRegion > iv) const noexcept=0;`	
Parameters (in):	data	the input data for the MAC
	expected	the expected most significant bits of the MAC digest
	iv	an optional initialization vector
Return value:	ara::core::Result< bool >	`true` if the calculated MAC equals the expected byte sequence, false otherwise.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kUninitializedContext	rollback_semantics
		if the context was not initialized by deploying a key
	ara::crypto::CryptoErrc::kUnsupported	rollback_semantics
		if single-call operation is not supported, or if the context requires setting an IV but no IV is provided, or if the context does not support setting an IV but an IV is provided
	ara::crypto::CryptoErrc::kInvalidInputSize	rollback_semantics
		if the size of the provided data, expected digest or IV is not supported
	ara::crypto::CryptoErrc::kBusyResource	rollback_semantics
		a required resource is busy
	ara::crypto::CryptoErrc::kEmptyKeySlot	rollback_semantics
		if a KeySlot was configured that is currently is empty
	ara::crypto::CryptoErrc::kUsageViolation	rollback_semantics
		if the allowed usage of the key does not include kAllowMac
Description:	Calculate a MAC digest and compare the most-significant bits of the calculated MAC digest against an expected value.	

⌋

8.20 Header: ara/crypto/cryp/protected_key_import_ctx.h

8.20.1 Class: ProtectedKeyImportCtx

[SWS_CRYPT_50084] Definition of API class ara::crypto::cryp::ProtectedKeyImportCtx

Upstream requirements: RS_CRYPTO_02105

⌈

Kind:	class
Header file:	#include "ara/crypto/cryp/protected_key_import_ctx.h"
Forwarding header file:	#include "ara/crypto/crypto_fwd.h"
Scope:	namespace ara::crypto::cryp
Symbol:	ProtectedKeyImportCtx
Base class:	CryptoContext
Syntax:	`class ProtectedKeyImportCtx : public CryptoContext {...};`
Description:	A contexts for importing secured data such as secret key-material.

⌋

8.20.1.1 Public Member Types

8.20.1.1.1 Type Alias: Sptr

[SWS_CRYPT_50085] Definition of API type ara::crypto::cryp::ProtectedKeyImportCtx::Sptr

Upstream requirements: RS_CRYPTO_02105

⌈

Kind:	type alias
Header file:	#include "ara/crypto/cryp/protected_key_import_ctx.h"
Scope:	`class ara::crypto::cryp::ProtectedKeyImportCtx`
Symbol:	Sptr
Syntax:	`using Sptr = std::shared_ptr<ProtectedKeyImportCtx>;`
Description:	Smart pointer of the interface.

⌋

8.20.1.2 Public Member Functions

8.20.1.2.1 Member Functions

8.20.1.2.1.1 GetCryptoObjectType

[SWS_CRYPT_50208] Definition of API function ara::crypto::cryp::ProtectedKeyImportCtx::GetCryptoObjectType

Upstream requirements: RS_CRYPTO_02105

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/protected_key_import_ctx.h"	
Scope:	`class ara::crypto::cryp::ProtectedKeyImportCtx`	
Syntax:	`virtual ara::core::Result< CryptoObjectType > GetCryptoObjectType () const noexcept=0;`	
Return value:	ara::core::Result< CryptoObjectType >	the CryptoObjectType of the last unwrapped CryptoObject
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kUnsupported	rollback_semantics
		if the context does not support identifying imported objects
	ara::crypto::CryptoErrc::kNoSuchValue	rollback_semantics
		if no CryptoObject has been imported yet
Description:	The interface shall return the type of the last successfully unwrapped CryptoObject.	

⌋

8.20.1.2.1.2 GetPrivateKey

[SWS_CRYPT_50209] Definition of API function ara::crypto::cryp::ProtectedKeyImportCtx::GetPrivateKey

Upstream requirements: RS_CRYPTO_02105

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/protected_key_import_ctx.h"	
Scope:	`class ara::crypto::cryp::ProtectedKeyImportCtx`	
Syntax:	`virtual ara::core::Result< PrivateKey::Sptr > GetPrivateKey () const noexcept=0;`	
Return value:	ara::core::Result< PrivateKey::Sptr >	a PrivateKey shared pointer
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kUnsupported	rollback_semantics
		if the context does not support importing private keys

▽

△

	ara::crypto::CryptoErrc::kNoSuchValue	rollback_semantics
		if the last imported CryptoObject is not a PrivateKey
	ara::crypto::CryptoErrc::kProcessingNotFinished	rollback_semantics
		if `Import` has not been executed successfully before
Description:	The interface shall return a shared pointer to the unwrapped PrivateKey.	

⌋

8.20.1.2.1.3 GetPublicKey

[SWS_CRYPT_50211] Definition of API function ara::crypto::cryp::ProtectedKeyImportCtx::GetPublicKey

Upstream requirements: RS_CRYPTO_02105

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/protected_key_import_ctx.h"	
Scope:	`class ara::crypto::cryp::ProtectedKeyImportCtx`	
Syntax:	`virtual ara::core::Result< PublicKey::Sptr > GetPublicKey () const noexcept=0;`	
Return value:	ara::core::Result< PublicKey::Sptr >	a PublicKey shared pointer
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kUnsupported	rollback_semantics
		if the context does not support importing public keys
	ara::crypto::CryptoErrc::kNoSuchValue	rollback_semantics
		if the last imported CryptoObject is not a PublicKey
	ara::crypto::CryptoErrc::kProcessingNotFinished	rollback_semantics
		if `Import` has not been executed successfully before
Description:	The interface shall return a shared pointer to the unwrapped PublicKey.	

⌋

8.20.1.2.1.4 GetResponse

[SWS_CRYPT_41020] Definition of API function ara::crypto::cryp::ProtectedKeyImportCtx::GetResponse

Upstream requirements: RS_CRYPTO_02105

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/protected_key_import_ctx.h"	
Scope:	`class ara::crypto::cryp::ProtectedKeyImportCtx`	
Syntax:	`virtual ara::core::Result< std::size_t > GetResponse (ReadWriteMem Region responseBuffer) const noexcept=0;`	
DIRECTION NOT DEFINED	responseBuffer	--
Return value:	ara::core::Result< std::size_t >	the number of Bytes written to the output buffer
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kUnsupported	rollback_semantics
		if the context does not provide a response
	ara::crypto::CryptoErrc::kInsufficientCapacity	rollback_semantics
		if the output buffer is too small
Description:	The interface shall write the response of the CryptoProvider to the last invocation of Import into the provided output buffer. E.g., this interface could return M4/M5 of a SHE LoadKey operation.	

⌋

8.20.1.2.1.5 GetSymmetricKey

[SWS_CRYPT_50210] Definition of API function ara::crypto::cryp::ProtectedKeyImportCtx::GetSymmetricKey

Upstream requirements: RS_CRYPTO_02105

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/protected_key_import_ctx.h"	
Scope:	`class ara::crypto::cryp::ProtectedKeyImportCtx`	
Syntax:	`virtual ara::core::Result< SymmetricKey::Sptr > GetSymmetricKey () const noexcept=0;`	
Return value:	ara::core::Result< SymmetricKey::Sptr >	a SymmetricKey shared pointer
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kUnsupported	rollback_semantics
		if the context does not support importing symmetric keys
	ara::crypto::CryptoErrc::kNoSuchValue	rollback_semantics
		if the last imported CryptoObject is not a SymmetricKey

▽

△

	ara::crypto::CryptoErrc::kProcessingNotFinished	rollback_semantics
		if `Import` has not been executed successfully before
Description:	The interface shall return a shared pointer to the unwrapped SymmetricKey.	

⌋

8.20.1.2.1.6 Import

[SWS_CRYPT_50207] Definition of API function ara::crypto::cryp::ProtectedKeyImportCtx::Import

Upstream requirements: RS_CRYPTO_02105

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/protected_key_import_ctx.h"	
Scope:	`class ara::crypto::cryp::ProtectedKeyImportCtx`	
Syntax:	`virtual ara::core::Result< void > Import (ReadOnlyMemRegion serialized, ara::core::Optional< WriteableKeySlot::Sptr > keySlot) noexcept=0;`	
Parameters (in):	serialized	the memory region that contains the serialized import data
	keySlot	an optional WriteableKeySlot to directly store the imported key
Return value:	ara::core::Result< void >	an empty result or an error
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kInvalidInputSize	rollback_semantics
		if `serialized` cannot be unwrapped due to an unexpected size
	ara::crypto::CryptoErrc::kInvalidArgument	rollback_semantics
		if `serialized` does not conform to the specified import format
	ara::crypto::CryptoErrc::kAuthTagNotValid	rollback_semantics
		if `serialized` cannot be authenticated (MAC or signature invalid)
	ara::crypto::CryptoErrc::kBusyResource	rollback_semantics
		if the operation temporarily cannot be performed
Description:	The interface shall unwrap the protected data provided, parse the content, and make the parsed content (key-material and meta-data) available for retrieval by the application. If optionally a Key Slot is specified, the unwrapped key material shall be directly serialized into the KeySlot.	

⌋

8.20.1.2.1.7 Reset

[SWS_CRYPT_50212] Definition of API function ara::crypto::cryp::ProtectedKeyImportCtx::Reset

Upstream requirements: RS_CRYPTO_02105

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/protected_key_import_ctx.h"	
Scope:	`class ara::crypto::cryp::ProtectedKeyImportCtx`	
Syntax:	`virtual ara::core::Result< void > Reset () const noexcept=0;`	
Return value:	ara::core::Result< void >	an empty result or an error
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kUnsupported	rollback_semantics
		if the context does not support resetting its' state.
Description:	The interface shall restore the context to its initial state by ensuring that no key-import-key is set and all previously imported data is deleted from the ProtectedKeyImportCtx's internal buffers.	

⌋

8.20.1.2.1.8 SetKey

[SWS_CRYPT_50206] Definition of API function ara::crypto::cryp::ProtectedKeyImportCtx::SetKey

Upstream requirements: RS_CRYPTO_02105

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/protected_key_import_ctx.h"	
Scope:	`class ara::crypto::cryp::ProtectedKeyImportCtx`	
Syntax:	`virtual ara::core::Result< void > SetKey (KeySlot::Sptr keySlot) noexcept=0;`	
Parameters (in):	keySlot	a keyslot containing a symmetric key used as transport key
Return value:	ara::core::Result< void >	an empty result or an error
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kUnsupported	rollback_semantics
		if the context only supports setting a SymmetricKey object
	ara::crypto::CryptoErrc::kIncompatibleObject	rollback_semantics
		if the key contained in the provided keySlot cannot be used as transport key
	ara::crypto::CryptoErrc::kUsageViolation	rollback_semantics
		if the allowed usage flags of the key contained in the provided Key Slot does not include kAllowKeyUnwrapping

▽

△

Description:	The interface shall configure the context to use the key stored in the provided KeySlot at the time the import operation is started as the transport key.

⌋

8.20.1.2.1.9 SetKey

[SWS_CRYPT_50205] Definition of API function ara::crypto::cryp::ProtectedKeyImportCtx::SetKey

Upstream requirements: RS_CRYPTO_02105

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/protected_key_import_ctx.h"	
Scope:	`class ara::crypto::cryp::ProtectedKeyImportCtx`	
Syntax:	`virtual ara::core::Result< void > SetKey (SymmetricKey::Sptr key) noexcept=0;`	
Parameters (in):	key	symmetric key object used as transport key
Return value:	ara::core::Result< void >	an empty result or an error
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kUnsupported	rollback_semantics
		if the context only supports setting a KeySlot
	ara::crypto::CryptoErrc::kIncompatibleObject	rollback_semantics
		if the provided key object cannot be used as transport key
	ara::crypto::CryptoErrc::kUsageViolation	rollback_semantics
		if the allowed usage flags of the key does not include kAllowKeyUnwrapping
Description:	The interface shall configure the context to use the provided symmetric key as transport key.	

⌋

8.21 Header: ara/crypto/cryp/random_generator_ctx.h

8.21.1 Class: RandomGeneratorCtx

[SWS_CRYPT_22900] Definition of API class ara::crypto::cryp::RandomGeneratorCtx

Upstream requirements: RS_CRYPTO_02206

⌈

Kind:	class
Header file:	#include "ara/crypto/cryp/random_generator_ctx.h"
Forwarding header file:	#include "ara/crypto/crypto_fwd.h"
Scope:	namespace ara::crypto::cryp
Symbol:	RandomGeneratorCtx
Base class:	CryptoContext
Syntax:	`class RandomGeneratorCtx : public CryptoContext {...};`
Description:	Interface of Random Number Generator Context.

⌋

8.21.1.1 Public Member Types

8.21.1.1.1 Type Alias: Sptr

[SWS_CRYPT_22901] Definition of API type ara::crypto::cryp::RandomGeneratorCtx::Sptr

Upstream requirements: RS_CRYPTO_02206

⌈

Kind:	type alias
Header file:	#include "ara/crypto/cryp/random_generator_ctx.h"
Scope:	`class ara::crypto::cryp::RandomGeneratorCtx`
Symbol:	Sptr
Syntax:	`using Sptr = std::shared_ptr<RandomGeneratorCtx>;`
Description:	Shared smart pointer of the interface.

⌋

8.21.1.2 Public Member Functions

8.21.1.2.1 Member Functions

8.21.1.2.1.1 AddEntropy

[SWS_CRYPT_22914] Definition of API function ara::crypto::cryp::RandomGeneratorCtx::AddEntropy

Upstream requirements: RS_CRYPTO_02206

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/random_generator_ctx.h"	
Scope:	`class ara::crypto::cryp::RandomGeneratorCtx`	
Syntax:	`virtual ara::core::Result< void > AddEntropy (ReadOnlyMemRegion entropy) noexcept=0;`	
Parameters (in):	entropy	a memory region with the additional entropy value
Return value:	ara::core::Result< void >	a result that is empty if the method is supported and the entropy has been added successfully, an error otherwise
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kUnsupported	rollback_semantics
		if the RandomGeneratorContext does not support adding entropy.
Description:	Update the internal state of the RNG by mixing it with the provided additional entropy.	

⌋

8.21.1.2.1.2 Generate

[SWS_CRYPT_22915] Definition of API function ara::crypto::cryp::RandomGeneratorCtx::Generate

Upstream requirements: RS_CRYPTO_02206

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/random_generator_ctx.h"	
Scope:	`class ara::crypto::cryp::RandomGeneratorCtx`	
Syntax:	`virtual ara::core::Result< void > Generate (ReadWriteMemRegion output) noexcept=0;`	
Parameters (out):	output	an output buffer the random sequence will be written into
Return value:	ara::core::Result< void >	an empty result on success or an error
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kUninitializedContext	rollback_semantics
		if this context has to be seeded by the application

▽

△

	ara::crypto::CryptoErrc::kBusyResource	rollback_semantics
		if this context is currently out-of-entropy and therefore cannot provide the requested number of random bytes
Description:	Fill the provided output buffer with a (pseudo)random sequence.	

⌋

8.21.1.2.1.3 Seed

[SWS_CRYPT_22911] Definition of API function ara::crypto::cryp::RandomGeneratorCtx::Seed

Upstream requirements: RS_CRYPTO_02206

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/random_generator_ctx.h"	
Scope:	`class ara::crypto::cryp::RandomGeneratorCtx`	
Syntax:	`virtual ara::core::Result< void > Seed (ReadOnlyMemRegion seed) noexcept=0;`	
Parameters (in):	seed	a memory region with the seed value
Return value:	ara::core::Result< void >	a result that is empty if the method is supported and the state has been set successfully, an error otherwise
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kUnsupported	rollback_semantics
		if the RandomGeneratorContext does not support seeding.
Description:	Set the internal state of the RNG using the provided seed.	

⌋

8.21.1.2.1.4 Seed

[SWS_CRYPT_22912] Definition of API function ara::crypto::cryp::RandomGeneratorCtx::Seed

Upstream requirements: RS_CRYPTO_02206

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/random_generator_ctx.h"	
Scope:	`class ara::crypto::cryp::RandomGeneratorCtx`	
Syntax:	`virtual ara::core::Result< void > Seed (SymmetricKey::Sptr seed) noexcept=0;`	
Parameters (in):	seed	a SymmetricKey with the key used as seed value

▽

△

Return value:	ara::core::Result< void >	a result that is empty if the method is supported and the state has been set successfully, an error otherwise
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kUsageViolation	rollback_semantics
		if the allowed usage of the key does not include kAllowRngInit
	ara::crypto::CryptoErrc::kUnsupported	rollback_semantics
		if the RandomGeneratorContext does not support seeding with a symmetric key.
Description:	Set the internal state of the RNG using the provided seed.	

⌋

8.22 Header: ara/crypto/cryp/signer_private_ctx.h

8.22.1 Class: SignerPrivateCtx

[SWS_CRYPT_23500] Definition of API class ara::crypto::cryp::SignerPrivateCtx

Upstream requirements: RS_CRYPTO_02204

⌈

Kind:	class
Header file:	#include "ara/crypto/cryp/signer_private_ctx.h"
Forwarding header file:	#include "ara/crypto/crypto_fwd.h"
Scope:	namespace ara::crypto::cryp
Symbol:	SignerPrivateCtx
Base class:	CryptoContext
Syntax:	`class SignerPrivateCtx : public CryptoContext {...};`
Description:	Signature Private key Context interface.

⌋

8.22.1.1 Public Member Types

8.22.1.1.1 Type Alias: Sptr

[SWS_CRYPT_23501] Definition of API type ara::crypto::cryp::SignerPrivateCtx::Sptr

Upstream requirements: RS_CRYPTO_02204

⌈

Kind:	type alias
Header file:	#include "ara/crypto/cryp/signer_private_ctx.h"
Scope:	`class ara::crypto::cryp::SignerPrivateCtx`
Symbol:	Sptr
Syntax:	`using Sptr = std::shared_ptr<SignerPrivateCtx>;`
Description:	Smart pointer of the interface.

⌋

8.22.1.2 Public Member Functions

8.22.1.2.1 Member Functions

8.22.1.2.1.1 Cancel

[SWS_CRYPT_50041] Definition of API function ara::crypto::cryp::SignerPrivateCtx::Cancel

Upstream requirements: RS_CRYPTO_02302

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/signer_private_ctx.h"	
Scope:	`class ara::crypto::cryp::SignerPrivateCtx`	
Syntax:	`virtual ara::core::Result< void > Cancel () noexcept=0;`	
Return value:	ara::core::Result< void >	an empty result if the ongoing streaming operation was cancelled successfully, or a result containing an error
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kProcessingNotStarted	rollback_semantics
		if data processing was not started by a call of the `Start()` method
Description:	Cancel the ongoing streaming operation that has been initiated by Start() and has not yet been finished.	

⌋

8.22.1.2.1.2 Finish

[SWS_CRYPT_50040] Definition of API function ara::crypto::cryp::SignerPrivateCtx::Finish

Upstream requirements: RS_CRYPTO_02302, RS_CRYPTO_02203

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/signer_private_ctx.h"	
Scope:	`class ara::crypto::cryp::SignerPrivateCtx`	
Syntax:	`virtual ara::core::Result< size_t > Finish (ReadWriteMemRegion signature, FormatId signatureFormat) noexcept=0;`	
Parameters (in):	signature	an output buffer that the signature will be written into
	signatureFormat	the format in which the signature will be written
Return value:	ara::core::Result< size_t >	a result containing the size of the output data or an error. The output data may be smaller than the size of the output buffer `signature`.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kProcessingNotStarted	rollback_semantics
		if the signature generation has not been started successfully.
	ara::crypto::CryptoErrc::kInsufficientCapacity	rollback_semantics
		if the output buffer is too small
	ara::crypto::CryptoErrc::kUnsupportedFormat	rollback_semantics
		if the specified FormatId is unknown or not supported
Description:	Finish the signature generation and write the signature to the provided output buffer.	

⌋

8.22.1.2.1.3 GetMaximumChunkSize

[SWS_CRYPT_50042] Definition of API function ara::crypto::cryp::SignerPrivateCtx::GetMaximumChunkSize

Upstream requirements: RS_CRYPTO_02309

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/signer_private_ctx.h"	
Scope:	`class ara::crypto::cryp::SignerPrivateCtx`	
Syntax:	`virtual ara::core::Result< std::size_t > GetMaximumChunkSize () const noexcept=0;`	
Return value:	ara::core::Result< std::size_t >	the maximum buffer size in bytes
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	This function does not specify any standardized errors.	

▽

△

Description:	Get the maximum input buffer size supported for processing in single calls to Update() or Sign() by this context.

⌋

8.22.1.2.1.4 GetSignatureSize

[SWS_CRYPT_29004] Definition of API function ara::crypto::cryp::SignerPrivateCtx::GetSignatureSize

Upstream requirements: RS_CRYPTO_02309

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/signer_private_ctx.h"	
Scope:	`class ara::crypto::cryp::SignerPrivateCtx`	
Syntax:	`virtual ara::core::Result< std::size_t > GetSignatureSize (FormatId signatureFormat) const noexcept=0;`	
Parameters (in):	signatureFormat	the format for which the size of the signature will be calculated
Return value:	ara::core::Result< std::size_t >	size of the signature value in bytes or an error
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kUnsupportedFormat	rollback_semantics
		if the specified FormatId is unknown or not supported
Description:	Get size of the signature value produced by the current algorithm in the provided format.	

⌋

8.22.1.2.1.5 SetKey

[SWS_CRYPT_50036] Definition of API function ara::crypto::cryp::SignerPrivateCtx::SetKey

Upstream requirements: RS_CRYPTO_02001, RS_CRYPTO_02003

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/signer_private_ctx.h"	
Scope:	`class ara::crypto::cryp::SignerPrivateCtx`	
Syntax:	`virtual ara::core::Result< void > SetKey (KeySlot::Sptr keySlot) noexcept=0;`	
Parameters (in):	keySlot	a keyslot containing a private key
Return value:	ara::core::Result< void >	An empty result or an error
Exception Safety:	exception safe	

▽

△

Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kIncompatibleObject	rollback_semantics
		if the provided keySlot is incompatible with this context
	ara::crypto::CryptoErrc::kUnsupported	rollback_semantics
		if the context only supports setting a PrivateKey
Description:	Configure the context to use the key stored in the provided KeySlot at the time the signature calculation is started.	

⌋

8.22.1.2.1.6 SetKey

[SWS_CRYPT_23515] Definition of API function ara::crypto::cryp::SignerPrivateCtx::SetKey

Upstream requirements: RS_CRYPTO_02001, RS_CRYPTO_02003

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/signer_private_ctx.h"	
Scope:	`class ara::crypto::cryp::SignerPrivateCtx`	
Syntax:	`virtual ara::core::Result< void > SetKey (PrivateKey::Sptr key) noexcept=0;`	
Parameters (in):	key	the source key object
Return value:	ara::core::Result< void >	An empty result or an error
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kIncompatibleObject	rollback_semantics
		if the provided key object is incompatible with this context
	ara::crypto::CryptoErrc::kUsageViolation	rollback_semantics
		if the allowed usage of the key does not include kAllowSig Generation
	ara::crypto::CryptoErrc::kUnsupported	rollback_semantics
		if the context only supports setting a KeySlot
Description:	Configure the context to use the provided key.	

⌋

8.22.1.2.1.7 Sign

[SWS_CRYPT_23512] Definition of API function ara::crypto::cryp::SignerPrivateCtx::Sign

Upstream requirements: RS_CRYPTO_02204

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/signer_private_ctx.h"	
Scope:	`class ara::crypto::cryp::SignerPrivateCtx`	
Syntax:	`virtual ara::core::Result< std::size_t > Sign (ReadOnlyMemRegion message, ReadWriteMemRegion signature, FormatId signatureFormat, ara::core::Optional< ReadOnlyMemRegion > context) const noexcept=0;`	
Parameters (in):	message	an input buffer containing the message to be signed
	signatureFormat	the format in which the signature will be written
	context	an optional "context" value
Parameters (out):	signature	an output buffer that the signature will be written into
Return value:	ara::core::Result< std::size_t >	a result containing the size of the output data or an error. The output data may be smaller than the size of the output buffer `signature`.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kUninitializedContext	rollback_semantics
		if the context was not initialized by a key value
	ara::crypto::CryptoErrc::kUnsupported	rollback_semantics
		if single-call processing is not supported, or if the context requires setting a context but no context is provided, or if the context does not support setting a context but a context is provided
	ara::crypto::CryptoErrc::kInsufficientCapacity	rollback_semantics
		if the output buffer is too small
	ara::crypto::CryptoErrc::kInvalidInputSize	rollback_semantics
		if the size of the input buffer is invalid
	ara::crypto::CryptoErrc::kBusyResource	rollback_semantics
		a required resource is busy
	ara::crypto::CryptoErrc::kEmptyKeySlot	rollback_semantics
		if a KeySlot was configured that is currently is empty
	ara::crypto::CryptoErrc::kUsageViolation	rollback_semantics
		if the allowed usage of the key does not include kAllowSig Generation
	ara::crypto::CryptoErrc::kUnsupportedFormat	rollback_semantics
		if the specified FormatId is unknown or not supported
Description:	This function signs a message.	

⌋

8.22.1.2.1.8 Start

[SWS_CRYPT_50038] Definition of API function ara::crypto::cryp::SignerPrivateCtx::Start

Upstream requirements: RS_CRYPTO_02302

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/signer_private_ctx.h"	
Scope:	`class ara::crypto::cryp::SignerPrivateCtx`	
Syntax:	`virtual ara::core::Result< void > Start (ReadOnlyMemRegion context) noexcept=0;`	
Parameters (in):	context	a context value
Return value:	ara::core::Result< void >	An empty result or an error
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kUninitializedContext	rollback_semantics
		if the context was not initialized by deploying a key
	ara::crypto::CryptoErrc::kUnsupported	rollback_semantics
		if the context does not support streaming or the algorithm does not support setting a context value
	ara::crypto::CryptoErrc::kBusyResource	rollback_semantics
		a required resource is busy
	ara::crypto::CryptoErrc::kEmptyKeySlot	rollback_semantics
		if a KeySlot was configured that is currently is empty
	ara::crypto::CryptoErrc::kUsageViolation	rollback_semantics
		if the allowed usage of the key does not include kAllowSigGeneration
	ara::crypto::CryptoErrc::kInvalidInputSize	rollback_semantics
		if the size of the context is invalid
Description:	Initialize the context for a new signature generation.	

⌋

8.22.1.2.1.9 Start

[SWS_CRYPT_50037] Definition of API function ara::crypto::cryp::SignerPrivateCtx::Start

Upstream requirements: RS_CRYPTO_02302

⌈

Kind:	function
Header file:	#include "ara/crypto/cryp/signer_private_ctx.h"
Scope:	`class ara::crypto::cryp::SignerPrivateCtx`
Syntax:	`virtual ara::core::Result< void > Start () noexcept=0;`

▽

△

Return value:	ara::core::Result< void >	An empty result or an error
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kUninitializedContext	rollback_semantics
		if the context was not initialized by deploying a key
	ara::crypto::CryptoErrc::kUnsupported	rollback_semantics
		if the context does not support streaming or the algorithm requires setting a context value
	ara::crypto::CryptoErrc::kBusyResource	rollback_semantics
		a required resource is busy
	ara::crypto::CryptoErrc::kEmptyKeySlot	rollback_semantics
		if a KeySlot was configured that is currently is empty
	ara::crypto::CryptoErrc::kUsageViolation	rollback_semantics
		if the allowed usage of the key does not include kAllowSigGeneration
Description:	Initialize the context for a new signature generation.	

⌋

8.22.1.2.1.10 Update

[SWS_CRYPT_50039] Definition of API function ara::crypto::cryp::SignerPrivateCtx::Update

Upstream requirements: RS_CRYPTO_02302

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/signer_private_ctx.h"	
Scope:	`class ara::crypto::cryp::SignerPrivateCtx`	
Syntax:	`virtual ara::core::Result< void > Update (ReadOnlyMemRegion data) noexcept=0;`	
Parameters (in):	data	a part of the input message that should be processed
Return value:	ara::core::Result< void >	An empty result or an error
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kProcessingNotStarted	rollback_semantics
		if the signature calculation was not initiated by a call of the `Start()` method
	ara::crypto::CryptoErrc::kInvalidInputSize	rollback_semantics
		if the size of provided data is not supported
Description:	Update the signature generation context by a new part of the message.	

⌋

8.23 Header: ara/crypto/cryp/symmetric_cipher_ctx.h

8.23.1 Class: SymmetricCipherCtx

[SWS_CRYPT_50136] Definition of API class ara::crypto::cryp::SymmetricCipherCtx

Upstream requirements: RS_CRYPTO_02201

⌈

Kind:	class
Header file:	#include "ara/crypto/cryp/symmetric_cipher_ctx.h"
Forwarding header file:	#include "ara/crypto/crypto_fwd.h"
Scope:	namespace ara::crypto::cryp
Symbol:	SymmetricCipherCtx
Base class:	CryptoContext
Syntax:	`class SymmetricCipherCtx : public CryptoContext {...};`
Description:	Interface for Symmetric Cipher Operations.

⌋

8.23.1.1 Public Member Types

8.23.1.1.1 Type Alias: Sptr

[SWS_CRYPT_50137] Definition of API type ara::crypto::cryp::SymmetricCipherCtx::Sptr

Upstream requirements: RS_CRYPTO_02201

⌈

Kind:	type alias
Header file:	#include "ara/crypto/cryp/symmetric_cipher_ctx.h"
Scope:	`class ara::crypto::cryp::SymmetricCipherCtx`
Symbol:	Sptr
Syntax:	`using Sptr = std::shared_ptr<SymmetricCipherCtx>;`
Description:	Smart pointer of the interface.

⌋

8.23.1.2 Public Member Functions

8.23.1.2.1 Member Functions

8.23.1.2.1.1 Cancel

[SWS_CRYPT_50046] Definition of API function ara::crypto::cryp::SymmetricCipherCtx::Cancel

Upstream requirements: RS_CRYPTO_02302

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/symmetric_cipher_ctx.h"	
Scope:	`class ara::crypto::cryp::SymmetricCipherCtx`	
Syntax:	`virtual ara::core::Result< void > Cancel () noexcept=0;`	
Return value:	ara::core::Result< void >	an empty result if the ongoing streaming operation was cancelled successfully, or a result containing an error
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kProcessingNotStarted	rollback_semantics
		if data processing was not started by a call of the `Start()` method
Description:	Cancel the ongoing streaming operation that has been initiated by Start() and has not yet been finished.	

⌋

8.23.1.2.1.2 EncryptSingleCallWithOutputIv

[SWS_CRYPT_50049] Definition of API function ara::crypto::cryp::SymmetricCipherCtx::EncryptSingleCallWithOutputIv

Upstream requirements: RS_CRYPTO_02302

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/symmetric_cipher_ctx.h"	
Scope:	`class ara::crypto::cryp::SymmetricCipherCtx`	
Syntax:	`virtual ara::core::Result< std::size_t > EncryptSingleCallWithOutputIv (ReadOnlyMemRegion in, ReadWriteMemRegion out, ReadWriteMemRegion ivOut) noexcept=0;`	
Parameters (in):	in	an input data buffer containing the ciphertext or plaintext
Parameters (out):	out	an output data buffer that the encrypted or decrypted data is written to
	ivOut	an output data buffer that the initialization vector (IV) generated by the context will be written to. If the algorithm does not use an IV, no data will be written into the buffer.
Return value:	ara::core::Result< std::size_t >	a result containing the size of the output data or an error. The output data may be smaller than the size of the output buffer `out`.
Exception Safety:	exception safe	

▽

△

Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kUninitializedContext	rollback_semantics
		if the context was not initialized by deploying a key
	ara::crypto::CryptoErrc::kUnsupported	rollback_semantics
		if single-call encryption is not supported, or if the algorithm requires providing an input IV
	ara::crypto::CryptoErrc::kInsufficientCapacity	rollback_semantics
		if the data or IV output buffer is too small
	ara::crypto::CryptoErrc::kInvalidInputSize	rollback_semantics
		if the size of the input buffer cannot be processed
	ara::crypto::CryptoErrc::kInvalidArgument	rollback_semantics
		if the provided input data contains invalid data
	ara::crypto::CryptoErrc::kBusyResource	rollback_semantics
		a required resource is busy
	ara::crypto::CryptoErrc::kEmptyKeySlot	rollback_semantics
		if a KeySlot was configured that is currently is empty
	ara::crypto::CryptoErrc::kUsageViolation	rollback_semantics
		if the allowed usage of the key does not include kAllowDataEncryption for encryption
	ara::crypto::CryptoErrc::kIncompatibleObject	rollback_semantics
		if type of the key is incompatible to the transformation or the transformation direction (encrypt)
Description:	Perform encryption in a single-call and return an IV generated by the context.	

⌋

8.23.1.2.1.3 Finish

[SWS_CRYPT_50045] Definition of API function ara::crypto::cryp::SymmetricCipherCtx::Finish

Upstream requirements: RS_CRYPTO_02302

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/symmetric_cipher_ctx.h"	
Scope:	`class ara::crypto::cryp::SymmetricCipherCtx`	
Syntax:	`virtual ara::core::Result< std::size_t > Finish (ReadOnlyMemRegion in, ReadWriteMemRegion out) noexcept=0;`	
Parameters (in):	in	an input data buffer containing a part of the ciphertext or plaintext
Parameters (out):	out	an output data buffer that contains encrypted or decrypted data. The required size of this buffer can be queried using GetRequiredOutputBufferSize()
Return value:	ara::core::Result< std::size_t >	a result containing the size of the output data or an error. The output data may be smaller than the size of the output buffer `out`.
Exception Safety:	exception safe	

▽

△

Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kProcessingNotStarted	rollback_semantics
		if data processing was not started by a call of the `Start()` method
	ara::crypto::CryptoErrc::kInsufficientCapacity	rollback_semantics
		if the output buffer is too small
	ara::crypto::CryptoErrc::kInvalidInputSize	rollback_semantics
		if the size of the input buffer cannot be processed
	ara::crypto::CryptoErrc::kInvalidArgument	rollback_semantics
		if the provided input data contains invalid data
	ara::crypto::CryptoErrc::kUnsupported	rollback_semantics
		if stream processing is not supported by the context or
Description:	Finalize the encryption/decryption without returning an IV.	

⌋

8.23.1.2.1.4 FinishEncrypt

[SWS_CRYPT_23618] Definition of API function ara::crypto::cryp::SymmetricCipherCtx::FinishEncrypt

Upstream requirements: RS_CRYPTO_02302

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/symmetric_cipher_ctx.h"	
Scope:	`class ara::crypto::cryp::SymmetricCipherCtx`	
Syntax:	`virtual ara::core::Result< std::size_t > FinishEncrypt (ReadOnlyMemRegion in, ReadWriteMemRegion out, ReadWriteMemRegion ivOut) noexcept=0;`	
Parameters (in):	in	an input data buffer containing a part of the ciphertext or plaintext
Parameters (out):	out	an output data buffer that contains encrypted or decrypted data
	ivOut	an output data buffer that contains the IV
Return value:	ara::core::Result< std::size_t >	a result containing the size of the output data or an error. The output data may be smaller than the size of the output buffer `out`.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kProcessingNotStarted	rollback_semantics
		if data processing was not started by a call of the `Start()` method
	ara::crypto::CryptoErrc::kInsufficientCapacity	rollback_semantics
		if the output buffer is too small
	ara::crypto::CryptoErrc::kInvalidInputSize	rollback_semantics
		if the size of the input buffer cannot be processed
	ara::crypto::CryptoErrc::kInvalidArgument	rollback_semantics
		if the provided input data contains invalid data

▽

△

	ara::crypto::CryptoErrc::kUnsupported	rollback_semantics
		if stream processing is not supported by the context or if this context does not use an IV or if the IV has been provided by the application in the Start() function
Description:	Finalize the encryption and return the IV generated by the context.	

⌋

8.23.1.2.1.5 GetMaximumChunkSize

[SWS_CRYPT_50050] Definition of API function ara::crypto::cryp::SymmetricCipherCtx::GetMaximumChunkSize

Upstream requirements: RS_CRYPTO_02309

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/symmetric_cipher_ctx.h"	
Scope:	`class ara::crypto::cryp::SymmetricCipherCtx`	
Syntax:	`virtual ara::core::Result< std::size_t > GetMaximumChunkSize () const noexcept=0;`	
Return value:	ara::core::Result< std::size_t >	the maximum buffer size in bytes
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	This function does not specify any standardized errors.	
Description:	Get the maximum input buffer size supported for processing in single calls to Update(), Finish(), FinishEncrypt(), or ProcessSingleCall() by this context.	

⌋

8.23.1.2.1.6 GetMaximumInputIvSize

[SWS_CRYPT_50142] Definition of API function ara::crypto::cryp::SymmetricCipherCtx::GetMaximumInputIvSize

Upstream requirements: RS_CRYPTO_02309

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/symmetric_cipher_ctx.h"	
Scope:	`class ara::crypto::cryp::SymmetricCipherCtx`	
Syntax:	`virtual ara::core::Result< std::size_t > GetMaximumInputIvSize (CryptoTransform transformDirection) const noexcept=0;`	
Parameters (in):	transformDirection	the transformation direction, i.e., encryption or decryption

▽

△

Return value:	ara::core::Result< std::size_t >	size of IV in bytes
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	This function does not specify any standardized errors.	
Description:	Get the maximum IV size required to be passed to the context in the functions Start() and ProcessSingleCallWithInputIv() for the given transformDirection.	

⌋

8.23.1.2.1.7 GetMinimumChunkSize

[SWS_CRYPT_50144] Definition of API function ara::crypto::cryp::SymmetricCipherCtx::GetMinimumChunkSize

Upstream requirements: RS_CRYPTO_02309

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/symmetric_cipher_ctx.h"	
Scope:	`class ara::crypto::cryp::SymmetricCipherCtx`	
Syntax:	`virtual ara::core::Result< std::size_t > GetMinimumChunkSize () const noexcept=0;`	
Return value:	ara::core::Result< std::size_t >	the minimum buffer size in bytes
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	This function does not specify any standardized errors.	
Description:	Get the minimum input buffer size supported for processing in single calls to Update(), Finish(), FinishEncrypt(), or ProcessSingleCall() by this context.	

⌋

8.23.1.2.1.8 GetMinimumInputIvSize

[SWS_CRYPT_50141] Definition of API function ara::crypto::cryp::SymmetricCipherCtx::GetMinimumInputIvSize

Upstream requirements: RS_CRYPTO_02309

⌈

Kind:	function
Header file:	#include "ara/crypto/cryp/symmetric_cipher_ctx.h"
Scope:	`class ara::crypto::cryp::SymmetricCipherCtx`
Syntax:	`virtual ara::core::Result< std::size_t > GetMinimumInputIvSize (CryptoTransform transformDirection) const noexcept=0;`

▽

△

Parameters (in):	transformDirection	the transformation direction, i.e., encryption or decryption
Return value:	ara::core::Result< std::size_t >	size of IV in bytes
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	This function does not specify any standardized errors.	
Description:	Get the minimum IV size required to be passed to the context in the functions Start() and ProcessSingleCallWithInputIv() for the given transformDirection.	

⌋

8.23.1.2.1.9 GetOutputIvSize

[SWS_CRYPT_50143] Definition of API function ara::crypto::cryp::SymmetricCipherCtx::GetOutputIvSize

Upstream requirements: RS_CRYPTO_02309

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/symmetric_cipher_ctx.h"	
Scope:	`class ara::crypto::cryp::SymmetricCipherCtx`	
Syntax:	`virtual ara::core::Result< std::size_t > GetOutputIvSize () const noexcept=0;`	
Return value:	ara::core::Result< std::size_t >	size of IV in bytes
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	This function does not specify any standardized errors.	
Description:	Get the output IV size that will be returned by the functions FinishEncrypt() (in case no IV has been provided by the application during the call to Start()), and EncryptSingleCallWithOutputIv().	

⌋

8.23.1.2.1.10 GetRequiredOutputBufferSize

[SWS_CRYPT_50140] Definition of API function ara::crypto::cryp::SymmetricCipherCtx::GetRequiredOutputBufferSize

Upstream requirements: RS_CRYPTO_02302

⌈

Kind:	function
Header file:	#include "ara/crypto/cryp/symmetric_cipher_ctx.h"
Scope:	`class ara::crypto::cryp::SymmetricCipherCtx`

▽

△

Syntax:	`virtual ara::core::Result< std::size_t > GetRequiredOutputBufferSize (std::size_t inputSize, bool isFinal) const noexcept;`	
Parameters (in):	inputSize	size of input data
	isFinal	if `true`, calculate maximum required output buffer size for Finish(), FinishEncrypt(), ProcessSingleCallWithInputIv(), and `EncryptSingleCallWithOutputIv()`, otherwise for Update()
Return value:	ara::core::Result< std::size_t >	a result containing the required capacity of the output buffer (in bytes) or an error
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kUninitializedContext	rollback_semantics
		if the context was not initialized by deploying a key and setting the transformation direction
Description:	Calculate the maximum required size of the output buffer depending on the input size in the current state of the context.	

⌋

8.23.1.2.1.11 ProcessSingleCall

[SWS_CRYPT_50047] Definition of API function ara::crypto::cryp::SymmetricCipherCtx::ProcessSingleCall

Upstream requirements: RS_CRYPTO_02302

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/symmetric_cipher_ctx.h"	
Scope:	`class ara::crypto::cryp::SymmetricCipherCtx`	
Syntax:	`virtual ara::core::Result< std::size_t > ProcessSingleCall (ReadOnlyMemRegion in, ReadWriteMemRegion out, CryptoTransform transformDirection) noexcept=0;`	
Parameters (in):	in	an input data buffer containing the ciphertext or plaintext
	transformDirection	the transformation direction, i.e., encryption or decryption
Parameters (out):	out	an output data buffer that the encrypted or decrypted data is written to
Return value:	ara::core::Result< std::size_t >	a result containing the size of the output data or an error. The output data may be smaller than the size of the output buffer `out`.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kUninitializedContext	rollback_semantics
		if the context was not initialized by deploying a key
	ara::crypto::CryptoErrc::kUnsupported	rollback_semantics
		if single-call en-/decryption is not supported, or if the context requires setting or returning an IV in the passed transformDirection
	ara::crypto::CryptoErrc::kInsufficientCapacity	rollback_semantics
		if the output buffer is too small

▽

△

	ara::crypto::CryptoErrc::kInvalidInputSize	rollback_semantics
		if the size of the input buffer cannot be processed
	ara::crypto::CryptoErrc::kInvalidArgument	rollback_semantics
		if the provided input data contains invalid data
	ara::crypto::CryptoErrc::kBusyResource	rollback_semantics
		a required resource is busy
	ara::crypto::CryptoErrc::kEmptyKeySlot	rollback_semantics
		if a KeySlot was configured that is currently is empty
	ara::crypto::CryptoErrc::kUsageViolation	rollback_semantics
		if the allowed usage of the key does not include kAllowDataEncryption for encryption, or kAllowDataDecryption for decryption, respectively
	ara::crypto::CryptoErrc::kIncompatibleObject	rollback_semantics
		if type of the key is incompatible to the transformation or the transformation direction
Description:	Perform en-/decryption in a single-call without setting or returning an IV.	

⌋

8.23.1.2.1.12 ProcessSingleCallWithInputIv

[SWS_CRYPT_50048] Definition of API function ara::crypto::cryp::SymmetricCipherCtx::ProcessSingleCallWithInputIv

Upstream requirements: RS_CRYPTO_02302

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/symmetric_cipher_ctx.h"	
Scope:	`class ara::crypto::cryp::SymmetricCipherCtx`	
Syntax:	`virtual ara::core::Result< std::size_t > ProcessSingleCallWithInputIv (ReadOnlyMemRegion in, ReadWriteMemRegion out, CryptoTransform transformDirection, ReadOnlyMemRegion iv) noexcept=0;`	
Parameters (in):	in	an input data buffer containing the ciphertext or plaintext
	transformDirection	the transformation direction, i.e., encryption or decryption
	iv	an input data buffer containing the initialization vector (IV) to be used by the context
Parameters (out):	out	an output data buffer that the encrypted or decrypted data is written to
Return value:	ara::core::Result< std::size_t >	a result containing the size of the output data or an error. The output data may be smaller than the size of the output buffer `out`.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kUninitializedContext	rollback_semantics
		if the context was not initialized by deploying a key
	ara::crypto::CryptoErrc::kUnsupported	rollback_semantics

▽

△

		if single-call en-/decryption is not supported, or if single-call processing with an input IV is not supported
	ara::crypto::CryptoErrc::kInsufficientCapacity	rollback_semantics
		if the data output buffer is too small
	ara::crypto::CryptoErrc::kInvalidInputSize	rollback_semantics
		if the size of the input buffer cannot be processed
	ara::crypto::CryptoErrc::kInvalidArgument	rollback_semantics
		if the provided input data contains invalid data
	ara::crypto::CryptoErrc::kBusyResource	rollback_semantics
		a required resource is busy
	ara::crypto::CryptoErrc::kEmptyKeySlot	rollback_semantics
		if a KeySlot was configured that is currently is empty
	ara::crypto::CryptoErrc::kUsageViolation	rollback_semantics
		if the allowed usage of the key does not include kAllowDataEncryption for encryption, or kAllowDataDecryption for decryption, respectively
	ara::crypto::CryptoErrc::kIncompatibleObject	rollback_semantics
		if type of the key is incompatible to the transformation or the transformation direction
Description:	Perform en-/decryption in a single-call, providing an IV to be used during en-/decryption.	

⌋

8.23.1.2.1.13 Seek

[SWS_CRYPT_23613] Definition of API function ara::crypto::cryp::SymmetricCipherCtx::Seek

Upstream requirements: RS_CRYPTO_02304

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/symmetric_cipher_ctx.h"	
Scope:	`class ara::crypto::cryp::SymmetricCipherCtx`	
Syntax:	`virtual ara::core::Result< void > Seek (std::int64_t offset, bool fromBegin) noexcept=0;`	
Parameters (in):	offset	the offset value in bytes, relative to begin or current position in the data stream
	fromBegin	the starting point for positioning within the stream: from begin (if `true`) or from current position (if `false`)
Return value:	ara::core::Result< void >	An empty result or an error
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kUnsupported	rollback_semantics
		if the seek operation is not supported by this context

▽

△

	ara::crypto::CryptoErrc::kProcessingNotStarted	rollback_semantics
		if the data processing was not started by a call of the `Start()` method
	ara::crypto::CryptoErrc::kOutOfBounds	rollback_semantics
		if the resulting offset value is too large or too small
	ara::crypto::CryptoErrc::kInvalidArgument	rollback_semantics
		if the resulting offset is not block aligned but the context requires block alignment
Description:	Set the position of the next byte within the data stream.	

⌋

8.23.1.2.1.14 SetKey

[SWS_CRYPT_50138] Definition of API function ara::crypto::cryp::SymmetricCipherCtx::SetKey

Upstream requirements: RS_CRYPTO_02001, RS_CRYPTO_02003

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/symmetric_cipher_ctx.h"	
Scope:	`class ara::crypto::cryp::SymmetricCipherCtx`	
Syntax:	`virtual ara::core::Result< void > SetKey (SymmetricKey::Sptr key) noexcept=0;`	
Parameters (in):	key	the SymmetricKey to be used.
Return value:	ara::core::Result< void >	An empty result or an error
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kUnsupported	rollback_semantics
		if the context only supports setting a KeySlot
	ara::crypto::CryptoErrc::kIncompatibleObject	rollback_semantics
		if the provided key object is incompatible with this context
Description:	Configure the context to use the provided symmetric key.	

⌋

8.23.1.2.1.15 SetKey

[SWS_CRYPT_50043] Definition of API function ara::crypto::cryp::SymmetricCipherCtx::SetKey

Upstream requirements: RS_CRYPTO_02001, RS_CRYPTO_02003

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/symmetric_cipher_ctx.h"	
Scope:	`class ara::crypto::cryp::SymmetricCipherCtx`	
Syntax:	`virtual ara::core::Result< void > SetKey (KeySlot::Sptr keySlot) noexcept=0;`	
Parameters (in):	keySlot	a keyslot containing a SymmetricKey.
Return value:	ara::core::Result< void >	An empty result or an error
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kIncompatibleObject	rollback_semantics
		if the provided keyslot is incompatible with this context
	ara::crypto::CryptoErrc::kUnsupported	rollback_semantics
		if the context only supports setting a SymmetricKey
Description:	Configure the context to use the key stored in the provided KeySlot at the time the en-/decryption is started.	

⌋

8.23.1.2.1.16 Start

[SWS_CRYPT_23625] Definition of API function ara::crypto::cryp::SymmetricCipherCtx::Start

Upstream requirements: RS_CRYPTO_02302

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/symmetric_cipher_ctx.h"	
Scope:	`class ara::crypto::cryp::SymmetricCipherCtx`	
Syntax:	`virtual ara::core::Result< void > Start (CryptoTransform transformDirection, ReadOnlyMemRegion iv) noexcept=0;`	
Parameters (in):	transformDirection	the transformation direction, i.e., encryption or decryption
	iv	an Initialization Vector (IV) or "nonce" value. The allowed size of the IV can be queried using the functions GetMinimumInputIvSize() and GetMaximumInputIvSize().
Return value:	ara::core::Result< void >	An empty result or an error
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kUninitializedContext	rollback_semantics
		if the context was not initialized by deploying a key

▽

△

	ara::crypto::CryptoErrc::kInvalidInputSize	rollback_semantics
		if the size of provided IV is not supported
	ara::crypto::CryptoErrc::kUnsupported	rollback_semantics
		if the context does not support setting an IV in the given transform Direction or if stream processing is not supported by the context
	ara::crypto::CryptoErrc::kBusyResource	rollback_semantics
		a required resource is busy
	ara::crypto::CryptoErrc::kEmptyKeySlot	rollback_semantics
		if a KeySlot was configured that is currently is empty
	ara::crypto::CryptoErrc::kUsageViolation	rollback_semantics
		if the allowed usage of the key does not include kAllowDataEncryption for encryption, or kAllowDataDecryption for decryption, respectively
	ara::crypto::CryptoErrc::kIncompatibleObject	rollback_semantics
		if type of the key is incompatible to the transformation or the transformation direction
Description:	Initialize the context for a new encryption or decryption and set the IV used by the context.	

⌋

8.23.1.2.1.17 Start

[SWS_CRYPT_50044] Definition of API function ara::crypto::cryp::SymmetricCipherCtx::Start

Upstream requirements: RS_CRYPTO_02302

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/symmetric_cipher_ctx.h"	
Scope:	`class ara::crypto::cryp::SymmetricCipherCtx`	
Syntax:	`virtual ara::core::Result< void > Start (CryptoTransform transform Direction) noexcept=0;`	
Parameters (in):	transformDirection	the transformation direction, i.e., encryption or decryption
Return value:	ara::core::Result< void >	An empty result or an error
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kUninitializedContext	rollback_semantics
		if the context was not initialized by deploying a key
	ara::crypto::CryptoErrc::kUnsupported	rollback_semantics
		if the context requires setting an IV in the given transformDirection or if stream processing is not supported by the context
	ara::crypto::CryptoErrc::kBusyResource	rollback_semantics
		a required resource is busy
	ara::crypto::CryptoErrc::kEmptyKeySlot	rollback_semantics
		if a KeySlot was configured that is currently is empty

▽

△

	ara::crypto::CryptoErrc::kUsageViolation	rollback_semantics
		if the allowed usage of the key does not include kAllowDataEncryption for encryption, or kAllowDataDecryption for decryption, respectively
	ara::crypto::CryptoErrc::kIncompatibleObject	rollback_semantics
		if type of the key is incompatible to the transformation or the transformation direction
Description:	Initialize the context for a new encryption or decryption. An IV is either not required for this context in this transformation direction, or set by the context itself.	

⌋

8.23.1.2.1.18 Update

[SWS_CRYPT_50139] Definition of API function ara::crypto::cryp::SymmetricCipherCtx::Update

Upstream requirements: RS_CRYPTO_02302

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/symmetric_cipher_ctx.h"	
Scope:	`class ara::crypto::cryp::SymmetricCipherCtx`	
Syntax:	`virtual ara::core::Result< std::size_t > Update (ReadOnlyMemRegion in, ReadWriteMemRegion out) noexcept=0;`	
Parameters (in):	in	an input data buffer containing a part of the ciphertext or plaintext
Parameters (out):	out	an output data buffer that the encrypted or decrypted data will be written into. The required size of this buffer can be queried using GetRequiredOutputBufferSize()
Return value:	ara::core::Result< std::size_t >	a result containing the size of the output data or an error. The output data may be smaller than the size of the output buffer out.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kProcessingNotStarted	rollback_semantics
		if data processing was not started by a call of the `Start()` method
	ara::crypto::CryptoErrc::kInsufficientCapacity	rollback_semantics
		if the output buffer is too small
	ara::crypto::CryptoErrc::kInvalidInputSize	rollback_semantics
		if the size of the input buffer is invalid
	ara::crypto::CryptoErrc::kUnsupported	rollback_semantics
		if stream processing is not supported by the context
Description:	Encrypt or decrypt a data buffer.	

⌋

8.24 Header: ara/crypto/cryp/symmetric_key_wrapper_ctx.h

8.24.1 Class: SymmetricKeyWrapperCtx

[SWS_CRYPT_24000] Definition of API class ara::crypto::cryp::SymmetricKeyWrapperCtx

Upstream requirements: RS_CRYPTO_02104, RS_CRYPTO_02208

⌈

Kind:	class
Header file:	#include "ara/crypto/cryp/symmetric_key_wrapper_ctx.h"
Forwarding header file:	#include "ara/crypto/crypto_fwd.h"
Scope:	namespace ara::crypto::cryp
Symbol:	SymmetricKeyWrapperCtx
Base class:	CryptoContext
Syntax:	`class SymmetricKeyWrapperCtx : public CryptoContext {...};`
Description:	Context of a symmetric key wrap algorithm.

⌋

8.24.1.1 Public Member Types

8.24.1.1.1 Type Alias: Sptr

[SWS_CRYPT_24001] Definition of API type ara::crypto::cryp::SymmetricKeyWrapperCtx::Sptr

Upstream requirements: RS_CRYPTO_02201

⌈

Kind:	type alias
Header file:	#include "ara/crypto/cryp/symmetric_key_wrapper_ctx.h"
Scope:	`class ara::crypto::cryp::SymmetricKeyWrapperCtx`
Symbol:	Sptr
Syntax:	`using Sptr = std::shared_ptr<SymmetricKeyWrapperCtx>;`
Description:	Smart pointer of the interface.

⌋

8.24.1.2 Public Member Functions

8.24.1.2.1 Member Functions

8.24.1.2.1.1 GetRequiredWrappedKeyBufferSize

[SWS_CRYPT_50035] Definition of API function ara::crypto::cryp::SymmetricKeyWrapperCtx::GetRequiredWrappedKeyBufferSize

Upstream requirements: RS_CRYPTO_02302

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/symmetric_key_wrapper_ctx.h"	
Scope:	`class ara::crypto::cryp::SymmetricKeyWrapperCtx`	
Syntax:	`virtual ara::core::Result< std::size_t > GetRequiredWrappedKeyBufferSize (const CryptoObject &key) const noexcept;`	
Parameters (in):	key	key to be wrapped
Return value:	ara::core::Result< std::size_t >	a result containing the required capacity of the output buffer in bytes or an error
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kUninitializedContext	rollback_semantics
		if the context was not initialized by deploying a key
Description:	Calculate the maximum required size of the wrapped key output buffer depending on the key to be wrapped.	

⌋

8.24.1.2.1.2 SetKey

[SWS_CRYPT_50034] Definition of API function ara::crypto::cryp::SymmetricKeyWrapperCtx::SetKey

Upstream requirements: RS_CRYPTO_02001, RS_CRYPTO_02003

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/symmetric_key_wrapper_ctx.h"	
Scope:	`class ara::crypto::cryp::SymmetricKeyWrapperCtx`	
Syntax:	`virtual ara::core::Result< void > SetKey (KeySlot::Sptr keySlot) noexcept=0;`	
Parameters (in):	keySlot	a keyslot containing a symmetric key
Return value:	ara::core::Result< void >	An empty result or an error
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kIncompatibleObject	rollback_semantics
		if the provided keySlot is incompatible with this context

▽

△

	ara::crypto::CryptoErrc::kUnsupported	rollback_semantics
		if the context only supports setting a SymmetricKey
Description:	Configure the context to use the key stored in the provided KeySlot at the time the key wrapping is performed.	

⌋

8.24.1.2.1.3 SetKey

[SWS_CRYPT_24018] Definition of API function ara::crypto::cryp::SymmetricKeyWrapperCtx::SetKey

Upstream requirements: RS_CRYPTO_02001, RS_CRYPTO_02003

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/symmetric_key_wrapper_ctx.h"	
Scope:	`class ara::crypto::cryp::SymmetricKeyWrapperCtx`	
Syntax:	`virtual ara::core::Result< void > SetKey (SymmetricKey::Sptr key) noexcept=0;`	
Parameters (in):	key	the the key encryption key (KEK) object
Return value:	ara::core::Result< void >	An empty result or an error
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kIncompatibleObject	rollback_semantics
		if the provided key object is incompatible with this symmetric key context
	ara::crypto::CryptoErrc::kUsageViolation	rollback_semantics
		if the allowed usage of the key does not include kAllowKeyWrapping or kAllowKeyUnwrapping
	ara::crypto::CryptoErrc::kUnsupported	rollback_semantics
		if the context only supports setting a KeySlot
Description:	Configure the context to use the provided key as key encryption key.	

⌋

8.24.1.2.1.4 UnwrapSymmetricKey

[SWS_CRYPT_24016] Definition of API function ara::crypto::cryp::SymmetricKeyWrapperCtx::UnwrapSymmetricKey

Upstream requirements: RS_CRYPTO_02115

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/symmetric_key_wrapper_ctx.h"	
Scope:	`class ara::crypto::cryp::SymmetricKeyWrapperCtx`	
Syntax:	`virtual ara::core::Result< SymmetricKey::Sptr > UnwrapSymmetricKey (ReadOnlyMemRegion wrappedKey, AlgId targetAlgId, AllowedUsageFlags allowedUsage) const noexcept=0;`	
Parameters (in):	wrappedKey	an input buffer that contains the wrapped key
	targetAlgId	the algId of the to be created unwrapped key
	allowedUsage	bit-flags that define a list of allowed transformations' types in which the unwrapped key can be used
Return value:	ara::core::Result< SymmetricKey::Sptr >	a result containing the unwrapped SymmetricKey object or an error
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kInvalidArgument	rollback_semantics
		if the wrapped key data cannot be unwrapped (e.g., invalid format or size)
	ara::crypto::CryptoErrc::kUninitializedContext	rollback_semantics
		if the context was not initialized by a key value
	ara::crypto::CryptoErrc::kUsageViolation	rollback_semantics
		if the allowed usage of the KEK does not include kAllowKeyUnwrapping
	ara::crypto::CryptoErrc::kUnknownIdentifier	rollback_semantics
		if `targetAlgId` has an unsupported value
	ara::crypto::CryptoErrc::kIncompatibleArguments	rollback_semantics
		if the `allowedUsage` is incompatible with the `targetAlgId` or the key type
	ara::crypto::CryptoErrc::kBusyResource	rollback_semantics
		a required resource is busy
	ara::crypto::CryptoErrc::kEmptyKeySlot	rollback_semantics
		if a KeySlot was configured that is currently is empty
	ara::crypto::CryptoErrc::kUsageViolation	rollback_semantics
		if the allowed usage of the KEK does not include kAllowKeyUnwrapping
Description:	Unwrap a wrapped key using the previously set key encryption key and return an SymmetricKey.	

⌋

8.24.1.2.1.5 Wrap

[SWS_CRYPT_24014] Definition of API function ara::crypto::cryp::SymmetricKeyWrapperCtx::Wrap

Upstream requirements: RS_CRYPTO_02201

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/symmetric_key_wrapper_ctx.h"	
Scope:	`class ara::crypto::cryp::SymmetricKeyWrapperCtx`	
Syntax:	`virtual ara::core::Result< std::size_t > Wrap (CryptoObject::Sptr key, ReadWriteMemRegion wrappedKey) const noexcept=0;`	
Parameters (in):	key	the key to be wrapped
Parameters (out):	wrappedKey	an output buffer that the wrapped key will be written to
Return value:	ara::core::Result< std::size_t >	a result containing either a the number of bytes of the wrapped key or an error
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kInvalidArgument	rollback_semantics
		if the `key` object cannot be wrapped (e.g., invalid size or type)
	ara::crypto::CryptoErrc::kUninitializedContext	rollback_semantics
		if the context was not initialized by a key value
	ara::crypto::CryptoErrc::kUsageViolation	rollback_semantics
		if the allowed usage of the KEK does not include kAllowKeyWrapping or the allowed usage of the object does not include kAllowSecuredExport
	ara::crypto::CryptoErrc::kInsufficientCapacity	rollback_semantics
		if the output buffer is too small
	ara::crypto::CryptoErrc::kBusyResource	rollback_semantics
		a required resource is busy
	ara::crypto::CryptoErrc::kEmptyKeySlot	rollback_semantics
		if a KeySlot was configured that is currently is empty
Description:	Wrap the provided key using the previously set key encryption key.	

⌋

8.25 Header: ara/crypto/cryp/verifier_public_ctx.h

8.25.1 Class: VerifierPublicCtx

[SWS_CRYPT_24100] Definition of API class ara::crypto::cryp::VerifierPublicCtx

Upstream requirements: RS_CRYPTO_02204

⌈

Kind:	class
Header file:	#include "ara/crypto/cryp/verifier_public_ctx.h"
Forwarding header file:	#include "ara/crypto/crypto_fwd.h"
Scope:	namespace ara::crypto::cryp
Symbol:	VerifierPublicCtx
Base class:	CryptoContext
Syntax:	`class VerifierPublicCtx : public CryptoContext {...};`
Description:	Signature Verification Public key Context interface.

⌋

8.25.1.1 Public Member Types

8.25.1.1.1 Type Alias: Sptr

[SWS_CRYPT_24101] Definition of API type ara::crypto::cryp::VerifierPublicCtx::Sptr

Upstream requirements: RS_CRYPTO_02204

⌈

Kind:	type alias
Header file:	#include "ara/crypto/cryp/verifier_public_ctx.h"
Scope:	`class ara::crypto::cryp::VerifierPublicCtx`
Symbol:	Sptr
Syntax:	`using Sptr = std::shared_ptr<VerifierPublicCtx>;`
Description:	Smart pointer of the interface.

⌋

8.25.1.2 Public Member Functions

8.25.1.2.1 Member Functions

8.25.1.2.1.1 Cancel

[SWS_CRYPT_50023] Definition of API function ara::crypto::cryp::VerifierPublicCtx::Cancel

Upstream requirements: RS_CRYPTO_02302

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/verifier_public_ctx.h"	
Scope:	`class ara::crypto::cryp::VerifierPublicCtx`	
Syntax:	`virtual ara::core::Result< void > Cancel () noexcept=0;`	
Return value:	ara::core::Result< void >	an empty result if the ongoing streaming operation was cancelled successfully, or a result containing an error
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kProcessingNotStarted	rollback_semantics
		if data processing was not started by a call of the `Start()` method
Description:	Cancel the ongoing streaming operation that has been initiated by Start() and has not yet been finished.	

⌋

8.25.1.2.1.2 Finish

[SWS_CRYPT_50022] Definition of API function ara::crypto::cryp::VerifierPublicCtx::Finish

Upstream requirements: RS_CRYPTO_02302, RS_CRYPTO_02203

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/verifier_public_ctx.h"	
Scope:	`class ara::crypto::cryp::VerifierPublicCtx`	
Syntax:	`virtual ara::core::Result< bool > Finish (ReadOnlyMemRegion signature, FormatId signatureFormat) noexcept=0;`	
Parameters (in):	signature	an input buffer containing the signature
	signatureFormat	the format in which the signature is provided
Return value:	ara::core::Result< bool >	An result containing `true` if the signature was verified successfully, `false` if the signature verification failed, or an error.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kProcessingNotStarted	rollback_semantics
		if the signature generation has not been started sucessfully.

▽

△

	ara::crypto::CryptoErrc::kInvalidInputSize	rollback_semantics
		if the size of the input buffer is invalid
	ara::crypto::CryptoErrc::kUnsupportedFormat	rollback_semantics
		if the specified format ID is not supported
	ara::crypto::CryptoErrc::kInvalidArgument	rollback_semantics
		if `signature` does not conform to the specified format
Description:	Finish the signature verification and return the verification result.	

⌋

8.25.1.2.1.3 GetMaximumChunkSize

[SWS_CRYPT_50025] Definition of API function ara::crypto::cryp::VerifierPublicCtx::GetMaximumChunkSize

Upstream requirements: RS_CRYPTO_02309

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/verifier_public_ctx.h"	
Scope:	`class ara::crypto::cryp::VerifierPublicCtx`	
Syntax:	`virtual std::size_t GetMaximumChunkSize () const noexcept=0;`	
Return value:	std::size_t	the maximum buffer size in bytes
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Get the maximum input buffer size supported for processing in single calls to Update() or Verify() by this context.	

⌋

8.25.1.2.1.4 SetKey

[SWS_CRYPT_50018] Definition of API function ara::crypto::cryp::VerifierPublicCtx::SetKey

Upstream requirements: RS_CRYPTO_02001, RS_CRYPTO_02003

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/verifier_public_ctx.h"	
Scope:	`class ara::crypto::cryp::VerifierPublicCtx`	
Syntax:	`virtual ara::core::Result< void > SetKey (KeySlot::Sptr keySlot) noexcept=0;`	
Parameters (in):	keySlot	a keyslot containing a public key

▽

△

Return value:	ara::core::Result< void >	An empty result or an error
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kIncompatibleObject	rollback_semantics
		if the provided keySlot is incompatible with this context
	ara::crypto::CryptoErrc::kUnsupported	rollback_semantics
		if the context only supports setting a PublicKey
Description:	Configure the context to use the key stored in the provided KeySlot at the time the signature verification is started.	

⌋

8.25.1.2.1.5 SetKey

[SWS_CRYPT_24115] Definition of API function ara::crypto::cryp::VerifierPublicCtx::SetKey

Upstream requirements: RS_CRYPTO_02001, RS_CRYPTO_02003

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/verifier_public_ctx.h"	
Scope:	`class ara::crypto::cryp::VerifierPublicCtx`	
Syntax:	`virtual ara::core::Result< void > SetKey (PublicKey::Sptr key) noexcept=0;`	
Parameters (in):	key	the source key object
Return value:	ara::core::Result< void >	An empty result or an error
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	CryptoErrc::kIncompatibleObject	rollback_semantics
		if the provided key object is incompatible with this signature verification context
	CryptoErrc::kUsageViolation	rollback_semantics
		if the allowed usage of the key does not include kAllowSigVerification
	ara::crypto::CryptoErrc::kUnsupported	rollback_semantics
		if the context only supports setting a KeySlot
Description:	Configure the context to use the provided key.	

⌋

8.25.1.2.1.6 Start

[SWS_CRYPT_50019] Definition of API function ara::crypto::cryp::VerifierPublicCtx::Start

Upstream requirements: RS_CRYPTO_02302

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/verifier_public_ctx.h"	
Scope:	`class ara::crypto::cryp::VerifierPublicCtx`	
Syntax:	`virtual ara::core::Result< void > Start () noexcept=0;`	
Return value:	ara::core::Result< void >	An empty result or an error
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kUninitializedContext	rollback_semantics
		if the context was not initialized by deploying a key
	ara::crypto::CryptoErrc::kUnsupported	rollback_semantics
		if the context does not support streaming or the algorithm requires setting a context value
	ara::crypto::CryptoErrc::kBusyResource	rollback_semantics
		a required resource is busy
	ara::crypto::CryptoErrc::kEmptyKeySlot	rollback_semantics
		if a KeySlot was configured that is currently is empty
	ara::crypto::CryptoErrc::kUsageViolation	rollback_semantics
		if the allowed usage of the key does not include kAllowSigVerification
Description:	Initialize the context for a new signature verification.	

⌋

8.25.1.2.1.7 Start

[SWS_CRYPT_50020] Definition of API function ara::crypto::cryp::VerifierPublicCtx::Start

Upstream requirements: RS_CRYPTO_02302

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/verifier_public_ctx.h"	
Scope:	`class ara::crypto::cryp::VerifierPublicCtx`	
Syntax:	`virtual ara::core::Result< void > Start (ReadOnlyMemRegion context) noexcept=0;`	
Parameters (in):	context	a context value
Return value:	ara::core::Result< void >	An empty result or an error
Exception Safety:	exception safe	

▽

△

Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kUninitializedContext	rollback_semantics
		if the context was not initialized by deploying a key
	ara::crypto::CryptoErrc::kUnsupported	rollback_semantics
		if the context does not support streaming or the algorithm does not support setting a context value
	ara::crypto::CryptoErrc::kBusyResource	rollback_semantics
		a required resource is busy
	ara::crypto::CryptoErrc::kEmptyKeySlot	rollback_semantics
		if a KeySlot was configured that is currently is empty
	ara::crypto::CryptoErrc::kUsageViolation	rollback_semantics
		if the allowed usage of the key does not include kAllowSig Verification
	ara::crypto::CryptoErrc::kInvalidInputSize	rollback_semantics
		if the size of the context is invalid
Description:	Initialize the context for a new signature verification.	

⌋

8.25.1.2.1.8 Update

[SWS_CRYPT_50021] Definition of API function ara::crypto::cryp::VerifierPublicCtx::Update

Upstream requirements: RS_CRYPTO_02302

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/verifier_public_ctx.h"	
Scope:	`class ara::crypto::cryp::VerifierPublicCtx`	
Syntax:	`virtual ara::core::Result< void > Update (ReadOnlyMemRegion data) noexcept=0;`	
Parameters (in):	data	a part of the input message that should be processed
Return value:	ara::core::Result< void >	An empty result or an error
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kProcessingNotStarted	rollback_semantics
		if the signature verification was not initiated by a call of the `Start()` method
	ara::crypto::CryptoErrc::kInvalidInputSize	rollback_semantics
		if the size of provided data is not supported
Description:	Update the signature verification context by a new part of the message.	

⌋

8.25.1.2.1.9 Verify

[SWS_CRYPT_24112] Definition of API function ara::crypto::cryp::VerifierPublicCtx::Verify

Upstream requirements: RS_CRYPTO_02204

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/verifier_public_ctx.h"	
Scope:	`class ara::crypto::cryp::VerifierPublicCtx`	
Syntax:	`virtual ara::core::Result< bool > Verify (ReadOnlyMemRegion message, ReadOnlyMemRegion signature, FormatId signatureFormat) const noexcept=0;`	
Parameters (in):	message	an input buffer containing the message
	signature	an input buffer containing the signature
	signatureFormat	the format in which the signature is provided
Return value:	ara::core::Result< bool >	An result containing `true` if the signature was verified successfully, `false` if the signature verification failed, or an error.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kUninitializedContext	rollback_semantics
		if the context was not initialized by a key value
	ara::crypto::CryptoErrc::kInvalidInputSize	rollback_semantics
		if the size of the input buffer is invalid
	ara::crypto::CryptoErrc::kUnsupported	rollback_semantics
		if the signature algorithm requires a context value
	ara::crypto::CryptoErrc::kUnsupportedFormat	rollback_semantics
		if the specified format ID is not supported
	ara::crypto::CryptoErrc::kInvalidArgument	rollback_semantics
		if `signature` does not conform to the specified format.
	ara::crypto::CryptoErrc::kBusyResource	rollback_semantics
		a required resource is busy
	ara::crypto::CryptoErrc::kEmptyKeySlot	rollback_semantics
		if a KeySlot was configured that is currently is empty
	ara::crypto::CryptoErrc::kUsageViolation	rollback_semantics
		if the allowed usage of the key does not include kAllowSig Verification
Description:	Verify the signature of the provided message in a single call.	

⌋

8.25.1.2.1.10 Verify

[SWS_CRYPT_50024] Definition of API function ara::crypto::cryp::VerifierPublicCtx::Verify

Upstream requirements: RS_CRYPTO_02204

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/verifier_public_ctx.h"	
Scope:	`class ara::crypto::cryp::VerifierPublicCtx`	
Syntax:	`virtual ara::core::Result< bool > Verify (ReadOnlyMemRegion message, ReadOnlyMemRegion context, ReadOnlyMemRegion signature, FormatId signatureFormat) const noexcept=0;`	
Parameters (in):	message	an input buffer containing the message
	context	a user supplied "context"
	signature	an input buffer containing the signature
	signatureFormat	the format in which the signature is provided
Return value:	ara::core::Result< bool >	An result containing `true` if the signature was verified successfully, `false` if the signature verification failed, or an error.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kUninitializedContext	rollback_semantics
		if the context was not initialized by a key value
	ara::crypto::CryptoErrc::kInvalidInputSize	rollback_semantics
		if the size of the input buffer is invalid
	ara::crypto::CryptoErrc::kUnsupported	rollback_semantics
		if the signature algorithm does not support a context value
	ara::crypto::CryptoErrc::kUnsupportedFormat	rollback_semantics
		if the specified format ID is not supported
	ara::crypto::CryptoErrc::kInvalidArgument	rollback_semantics
		if `signature` does not conform to the specified format
	ara::crypto::CryptoErrc::kBusyResource	rollback_semantics
		a required resource is busy
	ara::crypto::CryptoErrc::kEmptyKeySlot	rollback_semantics
		if a KeySlot was configured that is currently is empty
	ara::crypto::CryptoErrc::kUsageViolation	rollback_semantics
		if the allowed usage of the key does not include kAllowSig Verification
Description:	Verify the signature of the provided message with context in a single call.	

⌋

8.26 Header: ara/crypto/cryp/writeable_keyslot.h

8.26.1 Class: WriteableKeySlot

[SWS_CRYPT_50166] Definition of API class ara::crypto::cryp::WriteableKeySlot

Upstream requirements: RS_CRYPTO_02405

⌈

Kind:	class
Port Interfaces:	`CryptoKeySlotInterface`
Header file:	#include "ara/crypto/cryp/writeable_keyslot.h"
Forwarding header file:	#include "ara/crypto/crypto_fwd.h"
Scope:	namespace ara::crypto::cryp
Symbol:	WriteableKeySlot
Base class:	KeySlot
Syntax:	`class WriteableKeySlot : public KeySlot {...};`
Description:	WriteableKeySlot interface representing a writeable storage location for a key in a Crypto Provider.

⌋

8.26.1.1 Public Member Types

8.26.1.1.1 Type Alias: Sptr

[SWS_CRYPT_50172] Definition of API type ara::crypto::cryp::WriteableKeySlot::Sptr

Upstream requirements: RS_CRYPTO_02405

⌈

Kind:	type alias
Header file:	#include "ara/crypto/cryp/writeable_keyslot.h"
Scope:	`class ara::crypto::cryp::WriteableKeySlot`
Symbol:	Sptr
Syntax:	`using Sptr = std::shared_ptr<WriteableKeySlot>;`
Description:	Smart pointer of the interface.

⌋

8.26.1.2 Public Member Functions

8.26.1.2.1 Special Member Functions

8.26.1.2.1.1 Move Constructor

[SWS_CRYPT_50171] Definition of API function ara::crypto::cryp::WriteableKeySlot::WriteableKeySlot

Upstream requirements: RS_CRYPTO_02004

⌈

Kind:	function
Header file:	#include "ara/crypto/cryp/writeable_keyslot.h"
Scope:	`class ara::crypto::cryp::WriteableKeySlot`
Syntax:	`WriteableKeySlot (WriteableKeySlot &&)=delete;`
Description:	Move-Constructor.

⌋

8.26.1.2.1.2 Copy Constructor

[SWS_CRYPT_50170] Definition of API function ara::crypto::cryp::WriteableKeySlot::WriteableKeySlot

Upstream requirements: RS_CRYPTO_02004

⌈

Kind:	function
Header file:	#include "ara/crypto/cryp/writeable_keyslot.h"
Scope:	`class ara::crypto::cryp::WriteableKeySlot`
Syntax:	`WriteableKeySlot (const WriteableKeySlot &)=delete;`
Description:	Copy-Constructor.

⌋

8.26.1.2.1.3 Copy Assignment Operator

[SWS_CRYPT_50168] Definition of API function ara::crypto::cryp::WriteableKeySlot::operator=

Upstream requirements: RS_CRYPTO_02004

⌈

Kind:	function
Header file:	#include "ara/crypto/cryp/writeable_keyslot.h"
Scope:	`class ara::crypto::cryp::WriteableKeySlot`
Syntax:	`WriteableKeySlot & operator= (const WriteableKeySlot &other)=delete;`
Description:	Copy-assign another WriteableKeySlot to this instance.

⌋

8.26.1.2.1.4 Move Assignment Operator

[SWS_CRYPT_50169] Definition of API function ara::crypto::cryp::WriteableKeySlot::operator=

Upstream requirements: RS_CRYPTO_02004

⌈

Kind:	function
Header file:	#include "ara/crypto/cryp/writeable_keyslot.h"
Scope:	`class ara::crypto::cryp::WriteableKeySlot`
Syntax:	`WriteableKeySlot & operator= (WriteableKeySlot &&other)=delete;`
Description:	Move-assign another WriteableKeySlot to this instance.

⌋

8.26.1.2.1.5 Destructor

[SWS_CRYPT_50167] Definition of API function ara::crypto::cryp::WriteableKeySlot::~WriteableKeySlot

Upstream requirements: RS_CRYPTO_02405

⌈

Kind:	function
Header file:	#include "ara/crypto/cryp/writeable_keyslot.h"
Scope:	`class ara::crypto::cryp::WriteableKeySlot`
Syntax:	`virtual ~WriteableKeySlot () noexcept=default;`
Exception Safety:	exception safe
Description:	Destructor.

⌋

8.26.1.2.2 Member Functions

8.26.1.2.2.1 Clear

[SWS_CRYPT_30405] Definition of API function ara::crypto::cryp::WriteableKeySlot::Clear

Upstream requirements: RS_CRYPTO_02009

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/writeable_keyslot.h"	
Scope:	`class ara::crypto::cryp::WriteableKeySlot`	
Syntax:	`virtual ara::core::Result< void > Clear () const noexcept=0;`	
Return value:	ara::core::Result< void >	an empty result if the keyslot was already empty or successfully cleared
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kUnsupported	rollback_semantics
		if the KeySlot does not support clearing
	ara::crypto::CryptoErrc::kBusyResource	rollback_semantics
		if the operation temporarily cannot be performed
Description:	The interface shall delete the current content of the KeySlot.	

⌋

8.27 Header: ara/crypto/cryp/cryobj/crypto_object.h

8.27.1 Class: CryptoObject

[SWS_CRYPT_20500] Definition of API class ara::crypto::cryp::CryptoObject

Upstream requirements: RS_CRYPTO_02005

⌈

Kind:	class
Header file:	#include "ara/crypto/cryp/cryobj/crypto_object.h"
Forwarding header file:	#include "ara/crypto/crypto_fwd.h"
Scope:	namespace ara::crypto::cryp
Symbol:	CryptoObject
Base class:	ara::crypto::Serializable
Syntax:	`class CryptoObject : public ara::crypto::Serializable {...};`
Description:	A common interface for all cryptographic objects recognizable by the Crypto Provider.

⌋

8.27.1.1 Public Member Types

8.27.1.1.1 Type Alias: Sptr

[SWS_CRYPT_20501] Definition of API type ara::crypto::cryp::CryptoObject::Sptr

Upstream requirements: RS_CRYPTO_02005

⌈

Kind:	type alias
Header file:	#include "ara/crypto/cryp/cryobj/crypto_object.h"
Scope:	`class ara::crypto::cryp::CryptoObject`
Symbol:	Sptr
Syntax:	`using Sptr = std::shared_ptr<CryptoObject>;`
Description:	Smart pointer of the interface.

⌋

8.27.1.2 Public Member Functions

8.27.1.2.1 Member Functions

8.27.1.2.1.1 GetAlgId

[SWS_CRYPT_20505] Definition of API function ara::crypto::cryp::CryptoObject::GetAlgId

Upstream requirements: RS_CRYPTO_02005

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/cryobj/crypto_object.h"	
Scope:	`class ara::crypto::cryp::CryptoObject`	
Syntax:	`virtual ara::core::Result< ara::crypto::AlgIdString > GetAlgId () const noexcept=0;`	
Return value:	ara::core::Result< ara::crypto::AlgIdString >	A result containing the AlgId of this CryptoObject
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	This function does not specify any standardized errors.	
Description:	The interface shall return the AlgId of this CryptoObject.	

⌋

8.27.1.2.1.2 GetAllowedUsage

[SWS_CRYPT_50092] Definition of API function ara::crypto::cryp::CryptoObject::GetAllowedUsage

Upstream requirements: RS_CRYPTO_02005

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/cryobj/crypto_object.h"	
Scope:	`class ara::crypto::cryp::CryptoObject`	
Syntax:	`virtual ara::core::Result< AllowedUsageFlags > GetAllowedUsage () const noexcept=0;`	
Return value:	ara::core::Result< AllowedUsageFlags >	A result containing the allowed usage of the key
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	This function does not specify any standardized errors.	
Description:	The interface shall return the allowed usage of the key.	

⌋

8.27.1.2.1.3 Save

[SWS_CRYPT_20517] Definition of API function ara::crypto::cryp::CryptoObject::Save

Upstream requirements: RS_CRYPTO_02004

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/cryobj/crypto_object.h"	
Scope:	`class ara::crypto::cryp::CryptoObject`	
Syntax:	`virtual ara::core::Result< void > Save (WriteableKeySlot::Sptr keyslot) const noexcept=0;`	
Parameters (in):	keyslot	KeySlot representing the underlying storage
Return value:	ara::core::Result< void >	An empty result or an error
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kIncompatibleObject	rollback_semantics
		if the key is incompatible to the KeySlot, e.g., because the AlgId of the KeySlot is incompatible to the AlgId of the key or because the KeySlot belongs to a different CryptoProvider
	ara::crypto::CryptoErrc::kUsageViolation	rollback_semantics
		if the allowed usage of the key does not include kAllowPersist
	ara::crypto::CryptoErrc::kBusyResource	rollback_semantics
		if the operation temporarily cannot be performed

▽

△

Description:	The interface shall persistently store the key-material and its meta-data in the provided WriteableKeySlot. Data previously stored in keyslot shall be overwritten.

⌋

8.28 Header: ara/crypto/cryp/cryobj/private_key.h

8.28.1 Class: PrivateKey

[SWS_CRYPT_22500] Definition of API class ara::crypto::cryp::PrivateKey

Upstream requirements: RS_CRYPTO_02002, RS_CRYPTO_02403

⌈

Kind:	class
Header file:	#include "ara/crypto/cryp/cryobj/private_key.h"
Forwarding header file:	#include "ara/crypto/crypto_fwd.h"
Scope:	namespace ara::crypto::cryp
Symbol:	PrivateKey
Base class:	CryptoObject
Syntax:	`class PrivateKey : public CryptoObject {...};`
Description:	Generalized Asymmetric Private Key interface.

⌋

8.28.1.1 Public Member Types

8.28.1.1.1 Type Alias: Sptr

[SWS_CRYPT_22501] Definition of API type ara::crypto::cryp::PrivateKey::Sptr

Upstream requirements: RS_CRYPTO_02306

⌈

Kind:	type alias
Header file:	#include "ara/crypto/cryp/cryobj/private_key.h"
Scope:	`class ara::crypto::cryp::PrivateKey`
Symbol:	Sptr
Syntax:	`using Sptr = std::shared_ptr<PrivateKey>;`
Description:	shared smart pointer of the interface.

⌋

8.28.1.2 Public Member Functions

8.28.1.2.1 Member Functions

8.28.1.2.1.1 GetPublicKey

[SWS_CRYPT_22511] Definition of API function ara::crypto::cryp::PrivateKey::GetPublicKey

Upstream requirements: RS_CRYPTO_02108, RS_CRYPTO_02115

⌈

Kind:	function	
Header file:	#include "ara/crypto/cryp/cryobj/private_key.h"	
Scope:	`class ara::crypto::cryp::PrivateKey`	
Syntax:	`virtual ara::core::Result< PublicKey::Sptr > GetPublicKey () const noexcept=0;`	
Return value:	ara::core::Result< PublicKey::Sptr >	A result containing a shared pointer to the public key correspondent to this private key
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	This function does not specify any standardized errors.	
Description:	The interface shall return the public key correspondent to this private key.	

⌋

8.29 Header: ara/crypto/cryp/cryobj/public_key.h

8.29.1 Class: PublicKey

[SWS_CRYPT_22700] Definition of API class ara::crypto::cryp::PublicKey

Upstream requirements: RS_CRYPTO_02202

⌈

Kind:	class
Header file:	#include "ara/crypto/cryp/cryobj/public_key.h"
Forwarding header file:	#include "ara/crypto/crypto_fwd.h"
Scope:	namespace ara::crypto::cryp
Symbol:	PublicKey
Base class:	CryptoObject
Syntax:	`class PublicKey : public CryptoObject {...};`
Description:	General Asymmetric Public Key interface.

⌋

8.29.1.1 Public Member Types

8.29.1.1.1 Type Alias: Sptr

[SWS_CRYPT_22701] Definition of API type ara::crypto::cryp::PublicKey::Sptr

Upstream requirements: RS_CRYPTO_02202

⌈

Kind:	type alias
Header file:	#include "ara/crypto/cryp/cryobj/public_key.h"
Scope:	`class ara::crypto::cryp::PublicKey`
Symbol:	Sptr
Syntax:	`using Sptr = std::shared_ptr<PublicKey>;`
Description:	Shared smart pointer of the interface.

⌋

8.30 Header: ara/crypto/cryp/cryobj/symmetric_key.h

8.30.1 Class: SymmetricKey

[SWS_CRYPT_23800] Definition of API class ara::crypto::cryp::SymmetricKey

Upstream requirements: RS_CRYPTO_02001, RS_CRYPTO_02403

⌈

Kind:	class
Header file:	#include "ara/crypto/cryp/cryobj/symmetric_key.h"
Forwarding header file:	#include "ara/crypto/crypto_fwd.h"
Scope:	namespace ara::crypto::cryp
Symbol:	SymmetricKey
Base class:	CryptoObject
Syntax:	`class SymmetricKey : public CryptoObject {...};`
Description:	Symmetric Key interface.

⌋

8.30.1.1 Public Member Types

8.30.1.1.1 Type Alias: Sptr

[SWS_CRYPT_23801] Definition of API type ara::crypto::cryp::SymmetricKey::Sptr

Upstream requirements: RS_CRYPTO_02201

⌈

Kind:	type alias
Header file:	#include "ara/crypto/cryp/cryobj/symmetric_key.h"
Scope:	`class ara::crypto::cryp::SymmetricKey`
Symbol:	Sptr
Syntax:	`using Sptr = std::shared_ptr<SymmetricKey>;`
Description:	Shared smart pointer of the interface.

⌋

8.31 Header: ara/crypto/x509/cert_sign_request.h

8.31.1 Class: CertSignRequest

[SWS_CRYPT_40300] Definition of API class ara::crypto::x509::CertSignRequest

Upstream requirements: RS_CRYPTO_02306

⌈

Kind:	class
Header file:	#include "ara/crypto/x509/cert_sign_request.h"
Forwarding header file:	#include "ara/crypto/crypto_fwd.h"
Scope:	namespace ara::crypto::x509
Symbol:	CertSignRequest
Base class:	ara::crypto::Serializable
Syntax:	`class CertSignRequest : public ara::crypto::Serializable {...};`
Description:	Certificate Signing Request (CSR) object interface.

⌋

8.31.1.1 Public Member Types

8.31.1.1.1 Type Alias: Sptrc

[SWS_CRYPT_40301] Definition of API type ara::crypto::x509::CertSignRequest::Sptrc

Upstream requirements: RS_CRYPTO_02306

⌈

Kind:	type alias
Header file:	#include "ara/crypto/x509/cert_sign_request.h"
Scope:	`class ara::crypto::x509::CertSignRequest`
Symbol:	Sptrc
Syntax:	`using Sptrc = std::shared_ptr<const CertSignRequest>;`
Description:	Smart pointer of the constant interface.

⌋

8.32 Header: ara/crypto/x509/certificate.h

8.32.1 Non-Member Types

8.32.1.1 Type Alias: X509ExtendedKeyUsage

[SWS_CRYPT_50097] Definition of API type ara::crypto::x509::X509ExtendedKeyUsage

Upstream requirements: RS_CRYPTO_02306

⌈

Kind:	type alias
Header file:	#include "ara/crypto/x509/certificate.h"
Scope:	namespace ara::crypto::x509
Symbol:	X509ExtendedKeyUsage
Syntax:	`using X509ExtendedKeyUsage = std::uint32_t;`
Description:	X.509 v3 Extended Key Usage type definition.

⌋

8.32.1.2 Type Alias: X509KeyUsage

[SWS_CRYPT_40101] Definition of API type ara::crypto::x509::X509KeyUsage

Upstream requirements: RS_CRYPTO_02306

⌈

Kind:	type alias
Header file:	#include "ara/crypto/x509/certificate.h"
Scope:	namespace ara::crypto::x509
Symbol:	X509KeyUsage
Syntax:	`using X509KeyUsage = std::uint32_t;`
Description:	X.509 v3 Key Usage type definition.

⌋

8.32.2 Global Variables

8.32.2.1 kClientAuth

[SWS_CRYPT_50099] Definition of API variable ara::crypto::x509::kClientAuth

Upstream requirements: RS_CRYPTO_02306

⌈

Kind:	variable
Header file:	#include "ara/crypto/x509/certificate.h"
Scope:	namespace ara::crypto::x509
Symbol:	kClientAuth
Type:	X509ExtendedKeyUsage
Syntax:	`constexpr X509ExtendedKeyUsage kClientAuth {0x4000};`
Description:	TLS WWW client authentication.

⌋

8.32.2.2 kCodeSigning

[SWS_CRYPT_50100] Definition of API variable ara::crypto::x509::kCodeSigning

Upstream requirements: RS_CRYPTO_02306

⌈

Kind:	variable
Header file:	#include "ara/crypto/x509/certificate.h"
Scope:	namespace ara::crypto::x509
Symbol:	kCodeSigning

▽

△

Type:	X509ExtendedKeyUsage
Syntax:	`constexpr X509ExtendedKeyUsage kCodeSigning {0x2000};`
Description:	Signing of downloadable executable code.

⌋

8.32.2.3 kConstrCrlSign

[SWS_CRYPT_40157] Definition of API variable ara::crypto::x509::kConstrCrlSign

Upstream requirements: RS_CRYPTO_02306

⌈

Kind:	variable
Header file:	#include "ara/crypto/x509/certificate.h"
Scope:	namespace ara::crypto::x509
Symbol:	kConstrCrlSign
Type:	X509KeyUsage
Syntax:	`constexpr X509KeyUsage kConstrCrlSign {0x0200};`
Description:	The key can be used for Certificates Revokation Lists (CRL) signing.

⌋

8.32.2.4 kConstrDataEncipherment

[SWS_CRYPT_40154] Definition of API variable ara::crypto::x509::kConstrDataEncipherment

Upstream requirements: RS_CRYPTO_02306

⌈

Kind:	variable
Header file:	#include "ara/crypto/x509/certificate.h"
Scope:	namespace ara::crypto::x509
Symbol:	kConstrDataEncipherment
Type:	X509KeyUsage
Syntax:	`constexpr X509KeyUsage kConstrDataEncipherment {0x1000};`
Description:	The key can be used for data encipherment.

⌋

8.32.2.5 kConstrDecipherOnly

[SWS_CRYPT_40159] Definition of API variable ara::crypto::x509::kConstrDecipherOnly

Upstream requirements: RS_CRYPTO_02306

⌈

Kind:	variable
Header file:	#include "ara/crypto/x509/certificate.h"
Scope:	namespace ara::crypto::x509
Symbol:	kConstrDecipherOnly
Type:	X509KeyUsage
Syntax:	`constexpr X509KeyUsage kConstrDecipherOnly {0x0080};`
Description:	The enciphermet key can be used for deciphering only.

⌋

8.32.2.6 kConstrDigitalSignature

[SWS_CRYPT_40151] Definition of API variable ara::crypto::x509::kConstrDigitalSignature

Upstream requirements: RS_CRYPTO_02306

⌈

Kind:	variable
Header file:	#include "ara/crypto/x509/certificate.h"
Scope:	namespace ara::crypto::x509
Symbol:	kConstrDigitalSignature
Type:	X509KeyUsage
Syntax:	`constexpr X509KeyUsage kConstrDigitalSignature {0x8000};`
Description:	The key can be used for digital signature production.

⌋

8.32.2.7 kConstrEncipherOnly

[SWS_CRYPT_40158] Definition of API variable ara::crypto::x509::kConstrEncipherOnly

Upstream requirements: RS_CRYPTO_02306

⌈

Kind:	variable
Header file:	#include "ara/crypto/x509/certificate.h"
Scope:	namespace ara::crypto::x509
Symbol:	kConstrEncipherOnly
Type:	X509KeyUsage
Syntax:	`constexpr X509KeyUsage kConstrEncipherOnly {0x0100};`
Description:	The enciphermet key can be used for enciphering only.

⌋

8.32.2.8 kConstrKeyAgreement

[SWS_CRYPT_40155] Definition of API variable ara::crypto::x509::kConstrKeyAgreement

Upstream requirements: RS_CRYPTO_02306

⌈

Kind:	variable
Header file:	#include "ara/crypto/x509/certificate.h"
Scope:	namespace ara::crypto::x509
Symbol:	kConstrKeyAgreement
Type:	X509KeyUsage
Syntax:	`constexpr X509KeyUsage kConstrKeyAgreement {0x0800};`
Description:	The key can be used for a key agreement protocol execution.

⌋

8.32.2.9 kConstrKeyCertSign

[SWS_CRYPT_40156] Definition of API variable ara::crypto::x509::kConstrKeyCertSign

Upstream requirements: RS_CRYPTO_02306

⌈

Kind:	variable
Header file:	#include "ara/crypto/x509/certificate.h"
Scope:	namespace ara::crypto::x509
Symbol:	kConstrKeyCertSign
Type:	X509KeyUsage
Syntax:	`constexpr X509KeyUsage kConstrKeyCertSign {0x0400};`
Description:	The key can be used for certificates signing.

⌋

8.32.2.10 kConstrKeyEncipherment

[SWS_CRYPT_40153] Definition of API variable ara::crypto::x509::kConstrKeyEncipherment

Upstream requirements: RS_CRYPTO_02306

⌈

Kind:	variable
Header file:	#include "ara/crypto/x509/certificate.h"
Scope:	namespace ara::crypto::x509
Symbol:	kConstrKeyEncipherment
Type:	X509KeyUsage
Syntax:	`constexpr X509KeyUsage kConstrKeyEncipherment {0x2000};`
Description:	The key can be used for key encipherment.

⌋

8.32.2.11 kConstrNonRepudiation

[SWS_CRYPT_40152] Definition of API variable ara::crypto::x509::kConstrNonRepudiation

Upstream requirements: RS_CRYPTO_02306

⌈

Kind:	variable
Header file:	#include "ara/crypto/x509/certificate.h"
Scope:	namespace ara::crypto::x509
Symbol:	kConstrNonRepudiation
Type:	X509KeyUsage
Syntax:	`constexpr X509KeyUsage kConstrNonRepudiation {0x4000};`
Description:	The key can be used in cases requiring the "non-repudiation" guarantee.

⌋

8.32.2.12 kEmailProtection

[SWS_CRYPT_50101] Definition of API variable ara::crypto::x509::kEmailProtection

Upstream requirements: RS_CRYPTO_02306

⌈

Kind:	variable
Header file:	#include "ara/crypto/x509/certificate.h"
Scope:	namespace ara::crypto::x509
Symbol:	kEmailProtection
Type:	X509ExtendedKeyUsage
Syntax:	`constexpr X509ExtendedKeyUsage kEmailProtection {0x1000};`
Description:	Email protection.

⌋

8.32.2.13 kOCSPSigning

[SWS_CRYPT_50103] Definition of API variable ara::crypto::x509::kOCSPSigning

Upstream requirements: RS_CRYPTO_02306

⌈

Kind:	variable
Header file:	#include "ara/crypto/x509/certificate.h"
Scope:	namespace ara::crypto::x509
Symbol:	kOCSPSigning
Type:	X509ExtendedKeyUsage
Syntax:	`constexpr X509ExtendedKeyUsage kOCSPSigning {0x0400};`
Description:	Signing OCSP responses.

⌋

8.32.2.14 kServerAuth

[SWS_CRYPT_50098] Definition of API variable ara::crypto::x509::kServerAuth

Upstream requirements: RS_CRYPTO_02306

⌈

Kind:	variable
Header file:	#include "ara/crypto/x509/certificate.h"
Scope:	namespace ara::crypto::x509
Symbol:	kServerAuth
Type:	X509ExtendedKeyUsage
Syntax:	`constexpr X509ExtendedKeyUsage kServerAuth {0x8000};`
Description:	TLS WWW server authentication.

⌋

8.32.2.15 kTimeStamping

[SWS_CRYPT_50102] Definition of API variable ara::crypto::x509::kTimeStamping

Upstream requirements: RS_CRYPTO_02306

⌈

Kind:	variable
Header file:	#include "ara/crypto/x509/certificate.h"
Scope:	namespace ara::crypto::x509

▽

△

Symbol:	kTimeStamping
Type:	X509ExtendedKeyUsage
Syntax:	`constexpr X509ExtendedKeyUsage kTimeStamping {0x0800};`
Description:	Binding the hash of an object to a time.

⌋

8.32.3 Class: Certificate

[SWS_CRYPT_50180] Definition of API class ara::crypto::x509::Certificate

Upstream requirements: RS_CRYPTO_02306

⌈

Kind:	class
Header file:	#include "ara/crypto/x509/certificate.h"
Forwarding header file:	#include "ara/crypto/crypto_fwd.h"
Scope:	namespace ara::crypto::x509
Symbol:	Certificate
Base class:	ara::crypto::Serializable
Syntax:	`class Certificate :  public ara::crypto::Serializable {...};`
Description:	X.509 Certificate interface.

⌋

8.32.3.1 Public Member Types

8.32.3.1.1 Type Alias: Sptrc

[SWS_CRYPT_40202] Definition of API type ara::crypto::x509::Certificate::Sptrc

Upstream requirements: RS_CRYPTO_02306

⌈

Kind:	type alias
Header file:	#include "ara/crypto/x509/certificate.h"
Scope:	`class ara::crypto::x509::Certificate`
Symbol:	Sptrc
Syntax:	`using Sptrc = std::shared_ptr<const Certificate>;`
Description:	Smart pointer of the interface.

⌋

8.32.3.2 Public Member Functions

8.32.3.2.1 Member Functions

8.32.3.2.1.1 GetAuthorityInformationAccessLocation

[SWS_CRYPT_50214] Definition of API function ara::crypto::x509::Certificate::GetAuthorityInformationAccessLocation

Upstream requirements: RS_CRYPTO_02306

⌈

Kind:	function	
Header file:	#include "ara/crypto/x509/certificate.h"	
Scope:	`class ara::crypto::x509::Certificate`	
Syntax:	`virtual ara::core::Result< ara::core::Vector< ara::core::String > > GetAuthorityInformationAccessLocation () const noexcept=0;`	
Return value:	ara::core::Result< ara::core::Vector< ara::core::String > >	A result containing a vector of strings or an error.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kNoSuchValue	rollback_semantics
		if the certificate does not contain a Authority Information Access extension or the extension does not contain a accessLocation represented as a uniformResourceIdentifier IA5String
Description:	Get the Authority Information Access accessLocations represented as a uniformResource Identifier IA5String contained in the certificate according to the Authority Information Access extension.	

⌋

8.32.3.2.1.2 GetAuthorityKeyIdentifierExtension

[SWS_CRYPT_50181] Definition of API function ara::crypto::x509::Certificate::GetAuthorityKeyIdentifierExtension

Upstream requirements: RS_CRYPTO_02306

⌈

Kind:	function	
Header file:	#include "ara/crypto/x509/certificate.h"	
Scope:	`class ara::crypto::x509::Certificate`	
Syntax:	`virtual ara::core::Result< ara::core::Vector< std::uint8_t > > GetAuthorityKeyIdentifierExtension () const noexcept=0;`	
Return value:	ara::core::Result< ara::core::Vector< std::uint8_t > >	DER encoded AuthorityKeyIdentifier of this certificate
Exception Safety:	exception safe	
Thread Safety:	thread-safe	

▽

△

Errors:	ara::crypto::CryptoErrc::kNoSuchValue	rollback_semantics
		if the certificate does not contain an AuthorityKeyIdentifier
Description:	Get the DER encoded AuthorityKeyIdentifier of this certificate.	

⌋

8.32.3.2.1.3 GetCertificateSignatureAlgorithm

[SWS_CRYPT_50104] Definition of API function ara::crypto::x509::Certificate::GetCertificateSignatureAlgorithm

Upstream requirements: RS_CRYPTO_02306

⌈

Kind:	function	
Header file:	#include "ara/crypto/x509/certificate.h"	
Scope:	`class ara::crypto::x509::Certificate`	
Syntax:	`virtual ara::core::Result< AlgIdString > GetCertificateSignatureAlgorithm () const noexcept=0;`	
Return value:	ara::core::Result< AlgIdString >	the signature algorithm used by the CA to sign this certificate as an AlgId string
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	This function does not specify any standardized errors.	
Description:	Get the signature algorithm used by the CA to sign this certificate as an AlgId string.	

⌋

8.32.3.2.1.4 GetCrlDistributionPoints

[SWS_CRYPT_50213] Definition of API function ara::crypto::x509::Certificate::GetCrlDistributionPoints

Upstream requirements: RS_CRYPTO_02306

⌈

Kind:	function	
Header file:	#include "ara/crypto/x509/certificate.h"	
Scope:	`class ara::crypto::x509::Certificate`	
Syntax:	`virtual ara::core::Result< ara::core::Vector< ara::core::String > > GetCrlDistributionPoints () const noexcept=0;`	
Return value:	ara::core::Result< ara::core::Vector< ara::core::String > >	A result containing a vector of strings or an error.
Exception Safety:	exception safe	

▽

△

Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kNoSuchValue	rollback_semantics
		if the certificate does not contain a CRL Distribution Points extension or the extension does not contain a fullNames represented as a uniformResourceIdentifier IA5String
Description:	Get the CRL distribution point fullNames represented as a uniformResourceIdentifier IA5String contained in the certificate according to the CRL Distribution Points extension.	

⌋

8.32.3.2.1.5 GetExtendedKeyUsageExtension

[SWS_CRYPT_50106] Definition of API function ara::crypto::x509::Certificate::GetExtendedKeyUsageExtension

Upstream requirements: RS_CRYPTO_02306

⌈

Kind:	function	
Header file:	#include "ara/crypto/x509/certificate.h"	
Scope:	`class ara::crypto::x509::Certificate`	
Syntax:	`virtual ara::core::Result< X509ExtendedKeyUsage > GetExtendedKeyUsageExtension () const noexcept=0;`	
Return value:	ara::core::Result< X509ExtendedKeyUsage >	A result containing a bitmask of type X509ExtendedKeyUsage or an error.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kNoSuchValue	rollback_semantics
		if the certificate does not contain an extended key usage extension
Description:	Get the purpose of the key contained in the certificate according to the extended key usage extension.	

⌋

8.32.3.2.1.6 GetIssuer

[SWS_CRYPT_40213] Definition of API function ara::crypto::x509::Certificate::GetIssuer

Upstream requirements: RS_CRYPTO_02306

⌈

Kind:	function
Header file:	#include "ara/crypto/x509/certificate.h"
Scope:	`class ara::crypto::x509::Certificate`
Syntax:	`virtual ara::core::Result< X509DN::Sptrc > GetIssuer () const =0;`

▽

△

Return value:	ara::core::Result< X509 DN::Sptrc >	Issuer DN of this certificate
Exception Safety:	not exception safe	
Thread Safety:	thread-safe	
Errors:	This function does not specify any standardized errors.	
Description:	Get the issuer certificate DN.	

⌋

8.32.3.2.1.7 GetKeyUsageExtension

[SWS_CRYPT_40115] Definition of API function ara::crypto::x509::Certificate::GetKeyUsageExtension

Upstream requirements: RS_CRYPTO_02306

⌈

Kind:	function	
Header file:	#include "ara/crypto/x509/certificate.h"	
Scope:	`class ara::crypto::x509::Certificate`	
Syntax:	`virtual ara::core::Result< X509KeyUsage > GetKeyUsageExtension () const noexcept=0;`	
Return value:	ara::core::Result< X509 KeyUsage >	The KeyUsage containing the defined purposes of the key as a bitmask.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::k NoSuchValue	rollback_semantics
		if the certificate does not contain a key usage extension
Description:	Get the purpose of the key contained in the certificate according to the key usage extension.	

⌋

8.32.3.2.1.8 GetNotAfter

[SWS_CRYPT_40215] Definition of API function ara::crypto::x509::Certificate::GetNotAfter

Upstream requirements: RS_CRYPTO_02306

⌈

Kind:	function
Header file:	#include "ara/crypto/x509/certificate.h"
Scope:	`class ara::crypto::x509::Certificate`
Syntax:	`virtual ara::core::Result< Timestamp > GetNotAfter () const noexcept=0;`

▽

△

Return value:	ara::core::Result< Timestamp >	"Not After" of the certificate
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	This function does not specify any standardized errors.	
Description:	Get the "Not After" of the certificate.	

⌋

8.32.3.2.1.9 GetNotBefore

[SWS_CRYPT_40214] Definition of API function ara::crypto::x509::Certificate::GetNotBefore

Upstream requirements: RS_CRYPTO_02306

⌈

Kind:	function	
Header file:	#include "ara/crypto/x509/certificate.h"	
Scope:	`class ara::crypto::x509::Certificate`	
Syntax:	`virtual ara::core::Result< Timestamp > GetNotBefore () const noexcept=0;`	
Return value:	ara::core::Result< Timestamp >	"Not Before" of the certificate
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	This function does not specify any standardized errors.	
Description:	Get the "Not Before" of the certificate.	

⌋

8.32.3.2.1.10 GetSerialNumber

[SWS_CRYPT_40216] Definition of API function ara::crypto::x509::Certificate::GetSerialNumber

Upstream requirements: RS_CRYPTO_02306

⌈

Kind:	function
Header file:	#include "ara/crypto/x509/certificate.h"
Scope:	`class ara::crypto::x509::Certificate`
Syntax:	`virtual ara::core::Result< ara::core::Vector< std::uint8_t > > GetSerialNumber () const noexcept=0;`

▽

△

Return value:	ara::core::Result< ara::core::Vector< std::uint8_t > >	The serial number of this certificate.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	This function does not specify any standardized errors.	
Description:	Get the serial number of this certificate.	

⌋

8.32.3.2.1.11 GetSubject

[SWS_CRYPT_40112] Definition of API function ara::crypto::x509::Certificate::GetSubject

Upstream requirements: RS_CRYPTO_02306

⌈

Kind:	function	
Header file:	#include "ara/crypto/x509/certificate.h"	
Scope:	`class ara::crypto::x509::Certificate`	
Syntax:	`virtual ara::core::Result< X509DN::Sptrc > GetSubject () const noexcept=0;`	
Return value:	ara::core::Result< X509 DN::Sptrc >	subject DN
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	This function does not specify any standardized errors.	
Description:	Get the subject DN.	

⌋

8.32.3.2.1.12 GetSubjectKeyIdentifierExtension

[SWS_CRYPT_50182] Definition of API function ara::crypto::x509::Certificate::GetSubjectKeyIdentifierExtension

Upstream requirements: RS_CRYPTO_02306

⌈

Kind:	function
Header file:	#include "ara/crypto/x509/certificate.h"
Scope:	`class ara::crypto::x509::Certificate`
Syntax:	`virtual ara::core::Result< ara::core::Vector< std::uint8_t > > GetSubjectKeyIdentifierExtension () const noexcept=0;`

▽

△

Return value:	ara::core::Result< ara::core::Vector< std::uint8_t > >	DER encoded SubjectKeyIdentifier of this certificate
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kNoSuchValue	rollback_semantics
		if the certificate does not contain a SubjectKeyIdentifier
Description:	Get the DER encoded SubjectKeyIdentifier of this certificate.	

⌋

8.32.3.2.1.13 GetSubjectPublicKeyAlgorithm

[SWS_CRYPT_40111] Definition of API function ara::crypto::x509::Certificate::GetSubjectPublicKeyAlgorithm

Upstream requirements: RS_CRYPTO_02306

⌈

Kind:	function	
Header file:	#include "ara/crypto/x509/certificate.h"	
Scope:	`class ara::crypto::x509::Certificate`	
Syntax:	`virtual ara::core::Result< AlgIdString > GetSubjectPublicKeyAlgorithm () const noexcept=0;`	
Return value:	ara::core::Result< AlgId String >	the algorithm of the subject public key as an AlgId string
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	This function does not specify any standardized errors.	
Description:	Get the algorithm of the subject public key as an AlgId string.	

⌋

8.32.3.2.1.14 LoadSubjectPublicKey

[SWS_CRYPT_50105] Definition of API function ara::crypto::x509::Certificate::LoadSubjectPublicKey

Upstream requirements: RS_CRYPTO_02306

⌈

Kind:	function
Header file:	#include "ara/crypto/x509/certificate.h"
Scope:	`class ara::crypto::x509::Certificate`

▽

△

Syntax:	`virtual ara::core::Result< ara::crypto::cryp::PublicKey::Sptr > LoadSubjectPublicKey (ara::crypto::cryp::CryptoProvider::Sptr cryptoProvider) const noexcept=0;`	
Parameters (in):	cryptoProvider	the CryptoProvider to which the PublicKey will be loaded
Return value:	ara::core::Result< ara::crypto::cryp::PublicKey::Sptr >	the subject public key
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kIncompatibleArguments	rollback_semantics
		The CryptoProvider cannot load the PublicKey
	ara::crypto::CryptoErrc::kBusyResource	rollback_semantics
		if the operation temporarily cannot be performed
Description:	Load the subject public key as a PublicKey object usable in scope of the provided CryptoProvider. The allowed usage of the PublicKey will be set according to the key usage extension of the certificate.	

⌋

8.33 Header: ara/crypto/x509/certificate_validation_ctx.h

8.33.1 Non-Member Types

8.33.1.1 Type Alias: CertificateValidationOptions

[SWS_CRYPT_50107] Definition of API type ara::crypto::x509::CertificateValidationOptions

Upstream requirements: RS_CRYPTO_02306

⌈

Kind:	type alias
Header file:	#include "ara/crypto/x509/certificate_validation_ctx.h"
Scope:	namespace ara::crypto::x509
Symbol:	CertificateValidationOptions
Syntax:	`using CertificateValidationOptions = std::uint32_t;`
Description:	Type for holding bit-flags of certificate validation options.

⌋

8.33.2 Global Variables

8.33.2.1 kIgnoreCrl

[SWS_CRYPT_50111] Definition of API variable ara::crypto::x509::kIgnoreCrl

Upstream requirements: RS_CRYPTO_02306

⌈

Kind:	variable
Header file:	#include "ara/crypto/x509/certificate_validation_ctx.h"
Scope:	namespace ara::crypto::x509
Symbol:	kIgnoreCrl
Type:	CertificateValidationOptions
Syntax:	`constexpr CertificateValidationOptions kIgnoreCrl {0x0008};`
Description:	ignore previously imported CRLs

⌋

8.33.2.2 kIgnoreNotAfter

[SWS_CRYPT_50109] Definition of API variable ara::crypto::x509::kIgnoreNotAfter

Upstream requirements: RS_CRYPTO_02306

⌈

Kind:	variable
Header file:	#include "ara/crypto/x509/certificate_validation_ctx.h"
Scope:	namespace ara::crypto::x509
Symbol:	kIgnoreNotAfter
Type:	CertificateValidationOptions
Syntax:	`constexpr CertificateValidationOptions kIgnoreNotAfter {0x0002};`
Description:	ignore the NotAfter field

⌋

8.33.2.3 kIgnoreNotBefore

[SWS_CRYPT_50108] Definition of API variable ara::crypto::x509::kIgnoreNotBefore

Upstream requirements: RS_CRYPTO_02306

⌈

Kind:	variable
Header file:	#include "ara/crypto/x509/certificate_validation_ctx.h"
Scope:	namespace ara::crypto::x509
Symbol:	kIgnoreNotBefore
Type:	CertificateValidationOptions
Syntax:	`constexpr CertificateValidationOptions kIgnoreNotBefore {0x0001};`
Description:	ignore the NotBefore field

⌋

8.33.2.4 kIgnoreUnknownCriticalExtensions

[SWS_CRYPT_50110] Definition of API variable ara::crypto::x509::kIgnoreUnknownCriticalExtensions

Upstream requirements: RS_CRYPTO_02306

⌈

Kind:	variable
Header file:	#include "ara/crypto/x509/certificate_validation_ctx.h"
Scope:	namespace ara::crypto::x509
Symbol:	kIgnoreUnknownCriticalExtensions
Type:	CertificateValidationOptions
Syntax:	`constexpr CertificateValidationOptions kIgnoreUnknownCritical Extensions {0x0004};`
Description:	ignore all unknown critical extensions

⌋

8.33.2.5 kOcspOff

[SWS_CRYPT_50114] Definition of API variable ara::crypto::x509::kOcspOff

Upstream requirements: RS_CRYPTO_02306

⌈

Kind:	variable
Header file:	#include "ara/crypto/x509/certificate_validation_ctx.h"
Scope:	namespace ara::crypto::x509
Symbol:	kOcspOff
Type:	CertificateValidationOptions
Syntax:	`constexpr CertificateValidationOptions kOcspOff {0x0040};`
Description:	do not require OCSP response during certificate validation. This is the default behavior.

⌋

8.33.2.6 kOcspResponseCompleteChain

[SWS_CRYPT_50112] Definition of API variable ara::crypto::x509::kOcspResponseCompleteChain

Upstream requirements: RS_CRYPTO_02306

⌈

Kind:	variable
Header file:	#include "ara/crypto/x509/certificate_validation_ctx.h"
Scope:	namespace ara::crypto::x509
Symbol:	kOcspResponseCompleteChain
Type:	CertificateValidationOptions
Syntax:	`constexpr CertificateValidationOptions kOcspResponseCompleteChain {0x0010};`
Description:	require OCSP response for all certificates of chain

⌋

8.33.2.7 kOcspResponseTargetOnly

[SWS_CRYPT_50113] Definition of API variable ara::crypto::x509::kOcspResponseTargetOnly

Upstream requirements: RS_CRYPTO_02306

⌈

Kind:	variable
Header file:	#include "ara/crypto/x509/certificate_validation_ctx.h"
Scope:	namespace ara::crypto::x509
Symbol:	kOcspResponseTargetOnly
Type:	CertificateValidationOptions
Syntax:	`constexpr CertificateValidationOptions kOcspResponseTargetOnly {0x0020};`
Description:	require OCSP response only for target certificate

⌋

8.33.3 Class: CertificateValidationCtx

[SWS_CRYPT_50115] Definition of API class ara::crypto::x509::CertificateValidationCtx

Upstream requirements: RS_CRYPTO_02205

⌈

Kind:	class
Header file:	#include "ara/crypto/x509/certificate_validation_ctx.h"
Forwarding header file:	#include "ara/crypto/crypto_fwd.h"
Scope:	namespace ara::crypto::x509
Symbol:	CertificateValidationCtx
Syntax:	`class CertificateValidationCtx {...};`
Description:	Certificate Validation Context.

⌋

8.33.3.1 Public Member Types

8.33.3.1.1 Type Alias: Sptr

[SWS_CRYPT_50116] Definition of API type ara::crypto::x509::CertificateValidationCtx::Sptr

Upstream requirements: RS_CRYPTO_02205

⌈

Kind:	type alias
Header file:	#include "ara/crypto/x509/certificate_validation_ctx.h"
Scope:	`class ara::crypto::x509::CertificateValidationCtx`
Symbol:	Sptr
Syntax:	`using Sptr = std::shared_ptr<CertificateValidationCtx>;`
Description:	Smart pointer of the interface.

⌋

8.33.3.2 Public Member Functions

8.33.3.2.1 Member Functions

8.33.3.2.1.1 AddOcspResponse

[SWS_CRYPT_50124] Definition of API function ara::crypto::x509::CertificateValidationCtx::AddOcspResponse

Upstream requirements: RS_CRYPTO_02306

⌈

Kind:	function	
Header file:	#include "ara/crypto/x509/certificate_validation_ctx.h"	
Scope:	`class ara::crypto::x509::CertificateValidationCtx`	
Syntax:	`virtual ara::core::Result< CertificateValidationStatus > AddOcsp Response (OcspResponse::Sptrc ocspResponse, ara::core::Optional< Certificate::Sptrc > trustedOcspIssuerCertificate) noexcept=0;`	
Parameters (in):	ocspResponse	The OCSP response to be used during certificate validation.
	trustedOcspIssuer Certificate	If provided, the certificate to be considered trusted during validation of the OCSP response. If not provided, the certificates added via AddTrustedCertificate are considered trusted.
Return value:	ara::core::Result< CertificateValidation Status >	a result containing kValid if the response was added, or the result of the OCSP response validation, or an implementation specific error otherwise.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	This function does not specify any standardized errors.	

▽

△

Description:	Validate and add an OCSP response to be used during certificate validation to the set of previously added OCSP responses. Multiple OCSP responses may be added to one context. In order to validate the OCSP response, the issuer of the OCSP response must have been a) set as a trusted or untrusted certificate in this context previously, or b) the CRL must have been signed by another certificate that has the id-kp-OCSPSigning extended key usage set issued by the same CA, or c) the CRL must be signed by a trusted CA passed to this function using the optional trustedOcspIssuerCertificate parameter.

⌋

8.33.3.2.1.2 AddTrustedCertificate

[SWS_CRYPT_50117] Definition of API function ara::crypto::x509::CertificateValidationCtx::AddTrustedCertificate

Upstream requirements: RS_CRYPTO_02306

⌈

Kind:	function	
Header file:	#include "ara/crypto/x509/certificate_validation_ctx.h"	
Scope:	`class ara::crypto::x509::CertificateValidationCtx`	
Syntax:	`virtual ara::core::Result< void > AddTrustedCertificate ( Certificate::Sptrc trustedCertificate) noexcept=0;`	
Parameters (in):	trustedCertificate	a certificate to be considered trusted during certificate path validation.
Return value:	ara::core::Result< void >	An empty result on success, or an implementation specific error.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	This function does not specify any standardized errors.	
Description:	Add a trusted certificates to be used during certificate path validation to the set of previously added trusted certificates.	

⌋

8.33.3.2.1.3 AddUntrustedCertificate

[SWS_CRYPT_50118] Definition of API function ara::crypto::x509::CertificateValidationCtx::AddUntrustedCertificate

Upstream requirements: RS_CRYPTO_02306

⌈

Kind:	function
Header file:	#include "ara/crypto/x509/certificate_validation_ctx.h"
Scope:	`class ara::crypto::x509::CertificateValidationCtx`
Syntax:	`virtual ara::core::Result< void > AddUntrustedCertificate ( Certificate::Sptrc untrustedCertificate) noexcept=0;`

▽

△

Parameters (in):	untrustedCertificate	a certificate to be considered untrusted during certificate path validation.
Return value:	ara::core::Result< void >	An empty result on success, or an implementation specific error.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	This function does not specify any standardized errors.	
Description:	Add an untrusted certificate to be used during certificate path validation to the set of previously added untrusted certificates.	

⌋

8.33.3.2.1.4 IgnoreCriticalExtension

[SWS_CRYPT_50119] Definition of API function ara::crypto::x509::CertificateValidationCtx::IgnoreCriticalExtension

Upstream requirements: RS_CRYPTO_02306

⌈

Kind:	function	
Header file:	#include "ara/crypto/x509/certificate_validation_ctx.h"	
Scope:	`class ara::crypto::x509::CertificateValidationCtx`	
Syntax:	`virtual ara::core::Result< void > IgnoreCriticalExtension (ara::core::StringView ignoredExtensionOid) noexcept=0;`	
Parameters (in):	ignoredExtensionOid	the OID of the extension to be ignored during certificate and OCSP response validation
Return value:	ara::core::Result< void >	An empty result on success, or an implementation specific error.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	This function does not specify any standardized errors.	
Description:	Configure the context to ignore an extension during validation of certificates and OCSP responses, even if the extension is marked as critical and the validation context does not support the extension. It is the responsibility of the application to validate such extensions. Multiple invocations of this function lead to ignoring all passed extensions. This function has no effect if the kIgnoreUnknownCriticalExtensions validation option is set.	

⌋

8.33.3.2.1.5 SetCurrentTime

[SWS_CRYPT_50121] Definition of API function ara::crypto::x509::CertificateValidationCtx::SetCurrentTime

Upstream requirements: RS_CRYPTO_02306

⌈

Kind:	function	
Header file:	#include "ara/crypto/x509/certificate_validation_ctx.h"	
Scope:	`class ara::crypto::x509::CertificateValidationCtx`	
Syntax:	`virtual ara::core::Result< void > SetCurrentTime (Timestamp currentTime) noexcept=0;`	
Parameters (in):	currentTime	The time point to be used as the current time during certificate validation.
Return value:	ara::core::Result< void >	An empty result on success, or an implementation specific error.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	This function does not specify any standardized errors.	
Description:	Set the reference time point to be used as the current time during certificate validation. If this function is not called, the system time will be used during certificate validation.	

⌋

8.33.3.2.1.6 SetValidationOptions

[SWS_CRYPT_50120] Definition of API function ara::crypto::x509::CertificateValidationCtx::SetValidationOptions

Upstream requirements: RS_CRYPTO_02306

⌈

Kind:	function	
Header file:	#include "ara/crypto/x509/certificate_validation_ctx.h"	
Scope:	`class ara::crypto::x509::CertificateValidationCtx`	
Syntax:	`virtual ara::core::Result< void > SetValidationOptions (CertificateValidationOptions CertificateValidationOptions) noexcept=0;`	
Parameters (in):	CertificateValidationOptions	A bitmask identifying validation options. The semantics of each option is specified in the CertificateValidationOptions type.
Return value:	ara::core::Result< void >	An empty result on success, or an implementation specific error.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kInvalidArgument	rollback_semantics
		an invalid CertificateValidationOptions has been provided
Description:	Set validation options to be used during certificate validation.	

⌋

8.33.3.2.1.7 ValidateCertificate

[SWS_CRYPT_50122] Definition of API function ara::crypto::x509::CertificateValidationCtx::ValidateCertificate

Upstream requirements: RS_CRYPTO_02306

⌈

Kind:	function	
Header file:	#include "ara/crypto/x509/certificate_validation_ctx.h"	
Scope:	`class ara::crypto::x509::CertificateValidationCtx`	
Syntax:	`virtual ara::core::Result< CertificateValidationResult::Sptrc > ValidateCertificate (Certificate::Sptrc certificate) noexcept=0;`	
Parameters (in):	certificate	The certificate to be validated.
Return value:	ara::core::Result< CertificateValidation Result::Sptrc >	The result of the certificate validation, or an implementation specific error.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::k IncompatibleArguments	rollback_semantics
		if an OCSP response has been set via AddOcspResponse, but the CertificateValidationOptions do not specify the OCSP behavior.
Description:	Performs certificate path validation according to RFC 5280 and according to the previous configuration of the context.	

⌋

8.33.3.2.1.8 ValidateCertificateChain

[SWS_CRYPT_50123] Definition of API function ara::crypto::x509::CertificateValidationCtx::ValidateCertificateChain

Upstream requirements: RS_CRYPTO_02306

⌈

Kind:	function	
Header file:	#include "ara/crypto/x509/certificate_validation_ctx.h"	
Scope:	`class ara::crypto::x509::CertificateValidationCtx`	
Syntax:	`virtual ara::core::Result< CertificateValidationResult::Sptrc > ValidateCertificateChain (const ara::core::Span< Certificate::Sptrc > certificateChain) noexcept=0;`	
Parameters (in):	certificateChain	The certificate chain to be validated. Element 0 contains the leaf certificate and each subsequent certificate contains the issuer of the previous certificate.
Return value:	ara::core::Result< CertificateValidation Result::Sptrc >	The result of the certificate validation, or an implementation specific error.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	

▽

△

Errors:	ara::crypto::CryptoErrc::kIncompatibleArguments	rollback_semantics
		if an OCSP response has been set via AddOcspResponse, but the CertificateValidationOptions do not specify the OCSP behavior.
Description:	Performs certificate path validation according to RFC 5280 and according to the previous configuration of the context. The function ignores any certificates added to the context via Add UntrustedCertificate. Instead, the function validates the provided chain against any trusted certificate that has been set using the function AddTrustedCertificate.	

⌋

8.34 Header: ara/crypto/x509/certificate_validation_result.h

8.34.1 Non-Member Types

8.34.1.1 Enumeration: CertificateValidationStatus

[SWS_CRYPT_40203] Definition of API enum ara::crypto::x509::CertificateValidationStatus

Upstream requirements: RS_CRYPTO_02306

⌈

Kind:	enumeration	
Header file:	#include "ara/crypto/x509/certificate_validation_result.h"	
Forwarding header file:	#include "ara/crypto/crypto_fwd.h"	
Scope:	namespace ara::crypto::x509	
Symbol:	CertificateValidationStatus	
Underlying type:	std::uint8_t	
Syntax:	`enum class CertificateValidationStatus : std::uint8_t {...};`	
Values:	kValid	= 0
		The certificate path validation succeeded.
	kInvalidSignature	= 1
		The certificate path validation failed because a signature of a certificate in the path could not be verified.
	kNoPathToTrustedCertificate	= 3
		The certificate path validation failed because no path to a trusted certificate could be found.
	kExpired	= 4
		The certificate path validation failed because a certificate is expired according to the notAfter field.
	kFuture	= 5
		The certificate path validation failed because a certificate is not valid yet according to the notBefore field.
	kRevoked	= 6

▽

△

		The certificate path validation failed because a certificate has been revoked through a previously imported CRL or through an OCSP response (if OCSP is enabled through kOcspResponseComplete Chain or kOcspResponseTargetOnly)
	kUnknownOcspStatus	= 7
		The certificate path validation failed because a certificate state is 'unknown' according to an OCSP response (if OCSP is enabled through kOcspResponseCompleteChain or kOcspResponseTarget Only).
	kUnknownCritical Extension	= 8
		The certificate path validation failed because a certificate contains an unknown critical extension that has not been excluded from validation using the kIgnoreUnknownCriticalExtensions flag or the IgnoreCriticalExtension() function
Description:	Certificate Validation Status.	

⌋

8.34.2 Class: CertificateValidationResult

[SWS_CRYPT_50179] Definition of API class ara::crypto::x509::CertificateValidationResult

Upstream requirements: RS_CRYPTO_02306

⌈

Kind:	class
Header file:	#include "ara/crypto/x509/certificate_validation_result.h"
Forwarding header file:	#include "ara/crypto/crypto_fwd.h"
Scope:	namespace ara::crypto::x509
Symbol:	CertificateValidationResult
Syntax:	`class CertificateValidationResult {...};`
Description:	X.509 Certificate Validation Result.

⌋

8.34.2.1 Public Member Types

8.34.2.1.1 Type Alias: Sptrc

[SWS_CRYPT_50093] Definition of API type ara::crypto::x509::CertificateValidationResult::Sptrc

Upstream requirements: RS_CRYPTO_02306

⌈

Kind:	type alias
Header file:	#include "ara/crypto/x509/certificate_validation_result.h"
Scope:	`class ara::crypto::x509::CertificateValidationResult`
Symbol:	Sptrc
Syntax:	`using Sptrc = std::shared_ptr<const CertificateValidationResult>;`
Description:	Smart pointer of the interface.

⌋

8.34.2.2 Public Member Functions

8.34.2.2.1 Member Functions

8.34.2.2.1.1 GetCertificateChain

[SWS_CRYPT_50096] Definition of API function ara::crypto::x509::CertificateValidationResult::GetCertificateChain

Upstream requirements: RS_CRYPTO_02306

⌈

Kind:	function	
Header file:	#include "ara/crypto/x509/certificate_validation_result.h"	
Scope:	`class ara::crypto::x509::CertificateValidationResult`	
Syntax:	`virtual ara::core::Result< ara::core::Vector< Certificate::Sptrc > > GetCertificateChain () const noexcept=0;`	
Return value:	ara::core::Result< ara::core::Vector< Certificate::Sptrc > >	A vector containing the found certificate chain, beginning with the leaf certificate at position 0.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	This function does not specify any standardized errors.	
Description:	Get the certificate chain found during certificate validation.	

⌋

8.34.2.2.1.2 GetFailedCertificate

[SWS_CRYPT_50095] Definition of API function ara::crypto::x509::CertificateValidationResult::GetFailedCertificate

Upstream requirements: RS_CRYPTO_02306

⌈

Kind:	function	
Header file:	#include "ara/crypto/x509/certificate_validation_result.h"	
Scope:	`class ara::crypto::x509::CertificateValidationResult`	
Syntax:	`virtual ara::core::Result< Certificate::Sptrc > GetFailedCertificate () const noexcept=0;`	
Return value:	ara::core::Result< Certificate::Sptrc >	A result containing the certificate that caused the certificate path validation to fail
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kNoSuchValue	rollback_semantics
		if the certificate validation returned kValid, i.e., no certificate caused the validation to fail.
Description:	Get the certificate that caused the certificate path validation to fail.	

⌋

8.34.2.2.1.3 GetValidationStatus

[SWS_CRYPT_50094] Definition of API function ara::crypto::x509::CertificateValidationResult::GetValidationStatus

Upstream requirements: RS_CRYPTO_02306

⌈

Kind:	function	
Header file:	#include "ara/crypto/x509/certificate_validation_result.h"	
Scope:	`class ara::crypto::x509::CertificateValidationResult`	
Syntax:	`virtual ara::core::Result< CertificateValidationStatus > GetValidationStatus () const noexcept=0;`	
Return value:	ara::core::Result< CertificateValidationStatus >	The result of the certificate validation, or an implementation specific error.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	This function does not specify any standardized errors.	
Description:	Get the result of the certificate validation.	

⌋

8.35 Header: ara/crypto/x509/ocsp_response.h

8.35.1 Non-Member Types

8.35.1.1 Enumeration: OcspResponseStatus

[SWS_CRYPT_40001] Definition of API enum ara::crypto::x509::OcspResponseStatus

Upstream requirements: RS_CRYPTO_02306

⌈

Kind:	enumeration	
Header file:	#include "ara/crypto/x509/ocsp_response.h"	
Forwarding header file:	#include "ara/crypto/crypto_fwd.h"	
Scope:	namespace ara::crypto::x509	
Symbol:	OcspResponseStatus	
Underlying type:	std::uint8_t	
Syntax:	`enum class OcspResponseStatus : std::uint8_t {...};`	
Values:	kSuccessful	= 0
		Response has valid confirmations.
	kMalformedRequest	= 1
		Illegal confirmation request.
	kInternalError	= 2
		Internal error in issuer.
	kTryLater	= 3
		Try again later.
	kSigRequired	= 5
		Must sign the request.
	kUnauthorized	= 6
		Request unauthorized.
Description:	On-line Certificate Status Protocol (OCSP) Response Status.	

⌋

8.35.2 Class: OcspResponse

[SWS_CRYPT_50129] Definition of API class ara::crypto::x509::OcspResponse

Upstream requirements: RS_CRYPTO_02306

⌈

Kind:	class
Header file:	#include "ara/crypto/x509/ocsp_response.h"
Forwarding header file:	#include "ara/crypto/crypto_fwd.h"
Scope:	namespace ara::crypto::x509
Symbol:	OcspResponse

▽

△

Syntax:	`class OcspResponse {...};`
Description:	Interface of OcspResponse.

⌋

8.35.2.1 Public Member Types

8.35.2.1.1 Type Alias: Sptrc

[SWS_CRYPT_40802] Definition of API type ara::crypto::x509::OcspResponse::Sptrc

Upstream requirements: RS_CRYPTO_02306

⌈

Kind:	type alias
Header file:	#include "ara/crypto/x509/ocsp_response.h"
Scope:	`class ara::crypto::x509::OcspResponse`
Symbol:	Sptrc
Syntax:	`using Sptrc = std::shared_ptr<const OcspResponse>;`
Description:	Shared smart pointer of the interface.

⌋

8.35.2.2 Public Member Functions

8.35.2.2.1 Member Functions

8.35.2.2.1.1 GetCertificates

[SWS_CRYPT_50131] Definition of API function ara::crypto::x509::OcspResponse::GetCertificates

Upstream requirements: RS_CRYPTO_02306

⌈

Kind:	function	
Header file:	#include "ara/crypto/x509/ocsp_response.h"	
Scope:	`class ara::crypto::x509::OcspResponse`	
Syntax:	`virtual ara::core::Result< ara::core::Vector< Certificate::Sptrc > > GetCertificates () const noexcept=0;`	
Return value:	ara::core::Result< ara::core::Vector< Certificate::Sptrc > >	A Vector containing the certificates contained in the OCSP response.
Exception Safety:	exception safe	

▽

△

Thread Safety:	thread-safe
Errors:	This function does not specify any standardized errors.
Description:	Get the certificates contained in the OCSP response.

⌋

8.35.2.2.1.2 GetResponseStatus

[SWS_CRYPT_50130] Definition of API function ara::crypto::x509::OcspResponse::GetResponseStatus

Upstream requirements: RS_CRYPTO_02306

⌈

Kind:	function	
Header file:	#include "ara/crypto/x509/ocsp_response.h"	
Scope:	`class ara::crypto::x509::OcspResponse`	
Syntax:	`virtual ara::core::Result< OcspResponseStatus > GetResponseStatus () const noexcept=0;`	
Return value:	ara::core::Result< OcspResponseStatus >	The OCSP response status.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	This function does not specify any standardized errors.	
Description:	Get the OCSP response status.	

⌋

8.36 Header: ara/crypto/x509/x509_dn.h

8.36.1 Class: X509DN

[SWS_CRYPT_40400] Definition of API class ara::crypto::x509::X509DN

Upstream requirements: RS_CRYPTO_02306

⌈

Kind:	class
Header file:	#include "ara/crypto/x509/x509_dn.h"
Forwarding header file:	#include "ara/crypto/crypto_fwd.h"
Scope:	namespace ara::crypto::x509
Symbol:	X509DN
Base class:	ara::crypto::Serializable
Syntax:	`class X509DN : public ara::crypto::Serializable {...};`

▽

△

Description:	Interface of X.509 Distinguished Name (DN).

⌋

8.36.1.1 Public Member Types

8.36.1.1.1 Type Alias: Sptrc

[SWS_CRYPT_40401] Definition of API type ara::crypto::x509::X509DN::Sptrc

Upstream requirements: RS_CRYPTO_02306

⌈

Kind:	type alias
Header file:	#include "ara/crypto/x509/x509_dn.h"
Scope:	`class ara::crypto::x509::X509DN`
Symbol:	Sptrc
Syntax:	`using Sptrc = std::shared_ptr<const X509DN>;`
Description:	Const shared smart pointer of the interface.

⌋

8.36.1.2 Public Member Functions

8.36.1.2.1 Member Functions

8.36.1.2.1.1 ToString

[SWS_CRYPT_40411] Definition of API function ara::crypto::x509::X509DN::ToString

Upstream requirements: RS_CRYPTO_02306

⌈

Kind:	function	
Header file:	#include "ara/crypto/x509/x509_dn.h"	
Scope:	`class ara::crypto::x509::X509DN`	
Syntax:	`virtual ara::core::Result< ara::core::String > ToString () const noexcept=0;`	
Return value:	ara::core::Result< ara::core::String >	String of the whole DN string
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	This function does not specify any standardized errors.	

▽

△

Description:	Get the whole Distinguished Name (DN) as a single string according to RFC 4514. If the DN contains unstandardized attributes, the attribute's OID is used.

⌋

8.37 Header: ara/crypto/x509/x509_provider.h

8.37.1 Class: X509Provider

[SWS_CRYPT_40600] Definition of API class ara::crypto::x509::X509Provider

Upstream requirements: RS_CRYPTO_02306

⌈

Kind:	class
Port Interfaces:	`CryptoCertificateGroupInterface`, `CryptoCertificateInterface`
Header file:	#include "ara/crypto/x509/x509_provider.h"
Forwarding header file:	#include "ara/crypto/crypto_fwd.h"
Scope:	namespace ara::crypto::x509
Symbol:	X509Provider
Syntax:	`class X509Provider {...};`
Description:	X.509 Provider interface.

⌋

8.37.1.1 Public Member Types

8.37.1.1.1 Type Alias: Sptr

[SWS_CRYPT_40601] Definition of API type ara::crypto::x509::X509Provider::Sptr

Upstream requirements: RS_CRYPTO_02306

⌈

Kind:	type alias
Header file:	#include "ara/crypto/x509/x509_provider.h"
Scope:	`class ara::crypto::x509::X509Provider`
Symbol:	Sptr
Syntax:	`using Sptr = std::shared_ptr<X509Provider>;`
Description:	Smart pointer of the interface.

⌋

8.37.1.2 Public Member Functions

8.37.1.2.1 Special Member Functions

8.37.1.2.1.1 Move Constructor

[SWS_CRYPT_41014] Definition of API function ara::crypto::x509::X509Provider::X509Provider

Upstream requirements: RS_CRYPTO_02004

⌈

Kind:	function
Header file:	#include "ara/crypto/x509/x509_provider.h"
Scope:	`class ara::crypto::x509::X509Provider`
Syntax:	`X509Provider (X509Provider &&)=delete;`
Description:	Move-Constructor.

⌋

8.37.1.2.1.2 Copy Constructor

[SWS_CRYPT_41013] Definition of API function ara::crypto::x509::X509Provider::X509Provider

Upstream requirements: RS_CRYPTO_02004

⌈

Kind:	function
Header file:	#include "ara/crypto/x509/x509_provider.h"
Scope:	`class ara::crypto::x509::X509Provider`
Syntax:	`X509Provider (const X509Provider &)=delete;`
Description:	Copy-Constructor.

⌋

8.37.1.2.1.3 Move Assignment Operator

[SWS_CRYPT_30227] Definition of API function ara::crypto::x509::X509Provider::operator=

Upstream requirements: RS_CRYPTO_02004

⌈

Kind:	function
Header file:	#include "ara/crypto/x509/x509_provider.h"
Scope:	`class ara::crypto::x509::X509Provider`
Syntax:	`X509Provider & operator= (X509Provider &&other)=delete;`
Description:	Move-assign another X509Provider to this instance.

⌋

8.37.1.2.1.4 Copy Assignment Operator

[SWS_CRYPT_30226] Definition of API function ara::crypto::x509::X509Provider::operator=

Upstream requirements: RS_CRYPTO_02004

⌈

Kind:	function
Header file:	#include "ara/crypto/x509/x509_provider.h"
Scope:	`class ara::crypto::x509::X509Provider`
Syntax:	`X509Provider & operator= (const X509Provider &other)=delete;`
Description:	Copy-assign another X509Provider to this instance.

⌋

8.37.1.2.1.5 Destructor

[SWS_CRYPT_40604] Definition of API function ara::crypto::x509::X509Provider::~X509Provider

Upstream requirements: RS_CRYPTO_02306

⌈

Kind:	function
Header file:	#include "ara/crypto/x509/x509_provider.h"
Scope:	`class ara::crypto::x509::X509Provider`
Syntax:	`virtual ~X509Provider () noexcept=default;`
Exception Safety:	exception safe
Description:	Destructor.

⌋

8.37.1.2.2 Member Functions

8.37.1.2.2.1 ClearCertificateSlot

[SWS_CRYPT_40622] Definition of API function ara::crypto::x509::X509Provider::ClearCertificateSlot

Upstream requirements: RS_CRYPTO_02306

⌈

Kind:	function	
Header file:	#include "ara/crypto/x509/x509_provider.h"	
Scope:	`class ara::crypto::x509::X509Provider`	
Syntax:	`virtual ara::core::Result< bool > ClearCertificateSlot (const ara::core::InstanceSpecifier &instanceSpecifier) noexcept=0;`	
Parameters (in):	instanceSpecifier	An instance specifier identifying a storage location of a certificate.
Return value:	ara::core::Result< bool >	`true` if the slot contained a certificate which has been successfully removed `false` if the slot contained no certificate.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kBusyResource	rollback_semantics
		if the operation temporarily cannot be performed
Violations:	`ProcessMappingViolation`	In case InstanceSpecifier does not point to a PortPrototype typed by a CryptoCertificateInterface modeled for the current process.
	`InsufficientPermissionsViolation`	In case the current process does not have write permissions, i.e., CryptoCertificateToPortPrototypeMapping.writeAccess != True
Description:	Clear the certificate slot identified by the passed instance specifier.	

⌋

8.37.1.2.2.2 CreateCertSignRequest

[SWS_CRYPT_40640] Definition of API function ara::crypto::x509::X509Provider::CreateCertSignRequest

Upstream requirements: RS_CRYPTO_02306

⌈

Kind:	function	
Header file:	#include "ara/crypto/x509/x509_provider.h"	
Scope:	`class ara::crypto::x509::X509Provider`	
Syntax:	`virtual ara::core::Result< CertSignRequest::Sptrc > CreateCertSignRequest (cryp::SignerPrivateCtx::Sptr signerCtx, X509DN::Sptrc subjectDn, cryp::PublicKey::Sptr publicKey, ara::core::Optional< ReadOnlyMemRegion > attributes) const noexcept=0;`	
Parameters (in):	signerCtx	the fully configured SignerPrivateCtx to be used for signing this certificate request
	subjectDn	the DN to be used in the CSR
	publicKey	the public key to be used in the CSR. Must correspond to the private key configured in the signerCtx.

▽

△

	attributes	the DER-encoded attributes that should be included to the certificate signing request
Return value:	ara::core::Result< CertSignRequest::Sptrc >	Smart pointer to created certificate signing request
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kInvalidArgument	rollback_semantics
		if the attributes parameter contains invalid data
	ara::crypto::CryptoErrc::kUninitializedContext	rollback_semantics
		if the context was not initialized by a key value
	ara::crypto::CryptoErrc::kBusyResource	rollback_semantics
		a required resource is busy
	ara::crypto::CryptoErrc::kEmptyKeySlot	rollback_semantics
		if a KeySlot was configured that is currently is empty
	ara::crypto::CryptoErrc::kUsageViolation	rollback_semantics
		if the allowed usage of the key does not include kAllowSigGeneration
	ara::crypto::CryptoErrc::kInvalidArgument	rollback_semantics
		if the publicKey does not correspond to the private key configured at the signerCtx
Description:	Create certificate signing request according to RFC 2986.	

⌋

8.37.1.2.2.3 CreateCertificateValidationCtx

[SWS_CRYPT_50125] Definition of API function ara::crypto::x509::X509Provider::CreateCertificateValidationCtx

Upstream requirements: RS_CRYPTO_02306

⌈

Kind:	function	
Header file:	#include "ara/crypto/x509/x509_provider.h"	
Scope:	`class ara::crypto::x509::X509Provider`	
Syntax:	`virtual ara::core::Result< CertificateValidationCtx::Sptr > CreateCertificateValidationCtx () noexcept=0;`	
Return value:	ara::core::Result< CertificateValidationCtx::Sptr >	Smart pointer to the created context
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	This function does not specify any standardized errors.	
Description:	Create a certificate validation context.	

⌋

8.37.1.2.2.4 CreateDn

[SWS_CRYPT_50186] Definition of API function ara::crypto::x509::X509Provider::CreateDn

Upstream requirements: RS_CRYPTO_02306

⌈

Kind:	function	
Header file:	#include "ara/crypto/x509/x509_provider.h"	
Scope:	`class ara::crypto::x509::X509Provider`	
Syntax:	`virtual ara::core::Result< X509DN::Sptrc > CreateDn (ara::core::StringView dnString) const noexcept=0;`	
Parameters (in):	dnString	string representing a DN according to RFC 4514
Return value:	ara::core::Result< X509DN::Sptrc >	a result containing a X509DN or an error
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kInvalidArgument	rollback_semantics
		if the provided DN string is invalid.
Description:	Create DN from string representation.	

⌋

8.37.1.2.2.5 CreateDn

[SWS_CRYPT_50187] Definition of API function ara::crypto::x509::X509Provider::CreateDn

Upstream requirements: RS_CRYPTO_02306

⌈

Kind:	function	
Header file:	#include "ara/crypto/x509/x509_provider.h"	
Scope:	`class ara::crypto::x509::X509Provider`	
Syntax:	`virtual ara::core::Result< X509DN::Sptrc > CreateDn (ReadOnlyMemRegion dn, FormatId formatId) const noexcept=0;`	
Parameters (in):	dn	a DN serialized according to the provided format
	formatId	input format identifier
Return value:	ara::core::Result< X509DN::Sptrc >	a result containing a X509DN or an error
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kInvalidArgument	rollback_semantics
		if the provided dn buffer is invalid.
	ara::crypto::CryptoErrc::kUnsupportedFormat	rollback_semantics
		if the specified FormatId is unknown or not supported

▽

△

Description:	Create DN from a serialized buffer.

⌋

8.37.1.2.2.6 CreateOcspRequest

[SWS_CRYPT_40627] Definition of API function ara::crypto::x509::X509Provider::CreateOcspRequest

Upstream requirements: RS_CRYPTO_02306

⌈

Kind:	function	
Header file:	#include "ara/crypto/x509/x509_provider.h"	
Scope:	`class ara::crypto::x509::X509Provider`	
Syntax:	`virtual ara::core::Result< ara::core::Vector< std::uint8_t > > CreateOcspRequest (const ara::core::Span< Certificate::Sptrc > certList, const ara::core::Span< Certificate::Sptrc > issuerCertList, ara::core::Optional< cryp::SignerPrivateCtx::Sptr > signerCtx) noexcept=0;`	
Parameters (in):	certList	a list of certificates that should be verified
	issuerCertList	a list of issuer certificates of the certificates that should be verified
	signerCtx	an optional pointer to initialized signer context (if the request should be signed)
Return value:	ara::core::Result< ara::core::Vector< std::uint8_t > >	Smart pointer to the created OCSP request
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kUninitializedContext	rollback_semantics
		if the `signer` context is not initialized by a key
	ara::crypto::CryptoErrc::kBusyResource	rollback_semantics
		if the operation temporarily cannot be performed
Description:	Create OCSP request for specified list of certificates according to RFC 6960.	

⌋

8.37.1.2.2.7 CreateOcspRequest

[SWS_CRYPT_40626] Definition of API function ara::crypto::x509::X509Provider::CreateOcspRequest

Upstream requirements: RS_CRYPTO_02306

⌈

Kind:	function	
Header file:	#include "ara/crypto/x509/x509_provider.h"	
Scope:	`class ara::crypto::x509::X509Provider`	
Syntax:	`virtual ara::core::Result< ara::core::Vector< std::uint8_t > > CreateOcspRequest (Certificate::Sptrc cert, Certificate::Sptrc issuerCert, ara::core::Optional< cryp::SignerPrivateCtx::Sptr > signerCtx) noexcept=0;`	
Parameters (in):	cert	a certificate that should be verified
	issuerCert	the issuer certificate of the certificate that should be verified
	signerCtx	an optional pointer to initialized signer context (if the request should be signed)
Return value:	ara::core::Result< ara::core::Vector< std::uint8_t > >	Smart pointer to the created OCSP request
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kUninitializedContext	rollback_semantics
		if the `signer` context is not initialized by a key
	ara::crypto::CryptoErrc::kBusyResource	rollback_semantics
		if the operation temporarily cannot be performed
Description:	Create OCSP request for specified certificate according to RFC 6960.	

⌋

8.37.1.2.2.8 DeleteCrl

[SWS_CRYPT_50184] Definition of API function ara::crypto::x509::X509Provider::DeleteCrl

Upstream requirements: RS_CRYPTO_02306

⌈

Kind:	function	
Header file:	#include "ara/crypto/x509/x509_provider.h"	
Scope:	`class ara::crypto::x509::X509Provider`	
Syntax:	`virtual ara::core::Result< void > DeleteCrl (ara::core::InstanceSpecifier crlInstanceSpecifier) noexcept=0;`	
Parameters (in):	crlInstanceSpecifier	instance specifier to the RPortPrototype of a CryptoCertificateInterface that identifies the certificate for which the CRL shall be deleted
Return value:	ara::core::Result< void >	an empty result if no CRL was stored or if the CRL was successfully deleted, or a result containing an error

▽

△

Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kBusyResource	rollback_semantics
		if the operation temporarily cannot be performed
Violations:	`ProcessMappingViolation`	In case InstanceSpecifier does not point to a PortPrototype typed by a CryptoCertificateInterface modeled for the current process.
	`InsufficientPermissionsViolation`	In case the current process does not have write permissions, i.e., CryptoCertificateToPortPrototypeMapping.writeAccess != True
Description:	Delete a stored CRL identified by the passed instance specifier.	

⌋

8.37.1.2.2.9 ImportCertificate

[SWS_CRYPT_40621] Definition of API function ara::crypto::x509::X509Provider::ImportCertificate

Upstream requirements: RS_CRYPTO_02306

⌈

Kind:	function	
Header file:	#include "ara/crypto/x509/x509_provider.h"	
Scope:	`class ara::crypto::x509::X509Provider`	
Syntax:	`virtual ara::core::Result< void > ImportCertificate (const ara::core::InstanceSpecifier &instanceSpecifier, Certificate::Sptrc certificate) noexcept=0;`	
Parameters (in):	instanceSpecifier	an InstanceSpecifier identifying a certificate slot
	certificate	a certificate that should be imported
Return value:	ara::core::Result< void >	An empty result or an error
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kBusyResource	rollback_semantics
		if the operation temporarily cannot be performed
Violations:	`ProcessMappingViolation`	In case InstanceSpecifier does not point to a PortPrototype typed by a CryptoCertificateInterface modeled for the current process.
	`InsufficientPermissionsViolation`	In case the current process does not have write permissions, i.e., CryptoCertificateToPortPrototypeMapping.writeAccess != True
Description:	Persistently store a certificate in a certificate slot.	

⌋

8.37.1.2.2.10 ImportCertificateInGroup

[SWS_CRYPT_50127] Definition of API function ara::crypto::x509::X509Provider::ImportCertificateInGroup

Upstream requirements: RS_CRYPTO_02306

⌈

Kind:	function	
Header file:	#include "ara/crypto/x509/x509_provider.h"	
Scope:	`class ara::crypto::x509::X509Provider`	
Syntax:	`virtual ara::core::Result< void > ImportCertificateInGroup (const ara::core::InstanceSpecifier &instanceSpecifier, Certificate::Sptrc certificate) noexcept=0;`	
Parameters (in):	instanceSpecifier	an InstanceSpecifier identifying a certificate group
	certificate	a certificate that should be imported
Return value:	ara::core::Result< void >	An empty result or an error
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kInsufficientCapacity	rollback_semantics
		if the capacity of the certificate group is not sufficient to store the certificate
	ara::crypto::CryptoErrc::kBusyResource	rollback_semantics
		if the operation temporarily cannot be performed
Violations:	`ProcessMappingViolation`	In case InstanceSpecifier does not point to a PortPrototype typed by a CryptoCertificateGroupInterface modeled for the current process.
	`InsufficientPermissionsViolation`	In case the current process does not have write permissions, i.e., CryptoCertificateGroupToPortPrototypeMapping.writeAccess != True
Description:	Persistently store a certificate in a certificate group.	

⌋

8.37.1.2.2.11 ImportCrl

[SWS_CRYPT_50183] Definition of API function ara::crypto::x509::X509Provider::ImportCrl

Upstream requirements: RS_CRYPTO_02306

⌈

Kind:	function
Header file:	#include "ara/crypto/x509/x509_provider.h"
Scope:	`class ara::crypto::x509::X509Provider`
Syntax:	`virtual ara::core::Result< CertificateValidationStatus > ImportCrl (ara::core::InstanceSpecifier crlInstanceSpecifier, ReadOnlyMemRegion crl, FormatId formatId) noexcept=0;`

▽

△

Parameters (in):	crlInstanceSpecifier	`ara::core::InstanceSpecifier` to the `RPortPrototype` of a `CryptoCertificateInterface` that identifies the certificate for which the CRL shall be imported
	crl	serialized CRL or Delta CRL
	formatId	the format of the serialized CRL
Return value:	ara::core::Result< CertificateValidation Status >	a result containing kValid if the CRL was added, or the result of the CRL validation, or an error otherwise.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::k InvalidArgument	rollback_semantics
		if `crl` does not conform to the specified format
	ara::crypto::CryptoErrc::k UnsupportedFormat	rollback_semantics
		if the specified FormatId is unknown or not supported
	ara::crypto::CryptoErrc::k BusyResource	rollback_semantics
		if the operation temporarily cannot be performed
Violations:	`ProcessMappingViolation`	In case InstanceSpecifier does not point to a PortPrototype typed by a CryptoCertificateInterface modeled for the current process.
	`InsufficientPermissionsViolation`	In case the current process does not have write permissions, i.e., CryptoCertificateToPortPrototypeMapping.writeAccess != True
Description:	Parse and persist a Certificate Revocation List (CRL) or Delta CRL which is issued by the CA certificate identified by the passed `ara::core::InstanceSpecifier`.	

⌋

8.37.1.2.2.12 ParseCert

[SWS_CRYPT_40614] Definition of API function ara::crypto::x509::X509Provider::ParseCert

Upstream requirements: RS_CRYPTO_02306

⌈

Kind:	function	
Header file:	#include "ara/crypto/x509/x509_provider.h"	
Scope:	`class ara::crypto::x509::X509Provider`	
Syntax:	`virtual ara::core::Result< Certificate::Sptrc > ParseCert (ReadOnlyMemRegion cert, FormatId formatId) noexcept=0;`	
Parameters (in):	cert	certificate encoded in the given format
	formatId	input format identifier
Return value:	ara::core::Result< Certificate::Sptrc >	A result containing either the parsed certificate or an error.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::k InvalidArgument	rollback_semantics
		if the `cert` argument cannot be parsed
	ara::crypto::CryptoErrc::k UnsupportedFormat	rollback_semantics

▽

△

		if the specified FormatId is unknown or not supported
Description:	Parse a serialized representation of an X.509v3 certificate and create a certificate object.	

⌋

8.37.1.2.2.13 ParseCertChain

[SWS_CRYPT_40616] Definition of API function ara::crypto::x509::X509Provider::ParseCertChain

Upstream requirements: RS_CRYPTO_02306

⌈

Kind:	function	
Header file:	#include "ara/crypto/x509/x509_provider.h"	
Scope:	`class ara::crypto::x509::X509Provider`	
Syntax:	`virtual ara::core::Result< ara::core::Vector< Certificate::Sptrc > > ParseCertChain (ReadOnlyMemRegion certChain, FormatId formatId) noexcept=0;`	
Parameters (in):	certChain	certificate chain encoded in the given format
	formatId	input format identifier
Return value:	ara::core::Result< ara::core::Vector< Certificate::Sptrc > >	A result containing either the vector of certificates or an error.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kInvalidArgument	rollback_semantics
		if the `certChain` argument cannot be parsed
	ara::crypto::CryptoErrc::kUnsupportedFormat	rollback_semantics
		if the specified FormatId is unknown or not supported
Description:	Parse a serialized representation of a certificate chain, creating a new Certificate instance for each parsed certificate. Certificates in the returned vector will be placed in the same order as provided in certChain.	

⌋

8.37.1.2.2.14 ParseOcspResponse

[SWS_CRYPT_50185] Definition of API function ara::crypto::x509::X509Provider::ParseOcspResponse

Upstream requirements: RS_CRYPTO_02306

⌈

Kind:	function	
Header file:	#include "ara/crypto/x509/x509_provider.h"	
Scope:	`class ara::crypto::x509::X509Provider`	
Syntax:	`virtual ara::core::Result< OcspResponse::Sptrc > ParseOcspResponse (ReadOnlyMemRegion response, FormatId formatId) const noexcept=0;`	
Parameters (in):	response	a serialized OCSP response
	formatId	the format of the serialized OCSP response
Return value:	ara::core::Result< OcspResponse::Sptrc >	Smart pointer to the created OCSP response instance
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kInvalidArgument	rollback_semantics
		if the `response` argument cannot be parsed
	ara::crypto::CryptoErrc::kUnsupportedFormat	rollback_semantics
		if the specified FormatId is unknown or not supported
Description:	Parse serialized OCSP response.	

⌋

8.37.1.2.2.15 RemoveCertificateFromGroup

[SWS_CRYPT_50128] Definition of API function ara::crypto::x509::X509Provider::RemoveCertificateFromGroup

Upstream requirements: RS_CRYPTO_02306

⌈

Kind:	function	
Header file:	#include "ara/crypto/x509/x509_provider.h"	
Scope:	`class ara::crypto::x509::X509Provider`	
Syntax:	`virtual ara::core::Result< void > RemoveCertificateFromGroup (const ara::core::InstanceSpecifier &instanceSpecifier, Certificate::Sptrc certificate) noexcept=0;`	
Parameters (in):	instanceSpecifier	an InstanceSpecifier identifying a certificate group
	certificate	a certificate that should be removed from the certificate group
Return value:	ara::core::Result< void >	An empty result or an error
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kNoSuchValue	rollback_semantics

▽

△

		if the certificate group does not contain the certificate to be removed
	ara::crypto::CryptoErrc::kBusyResource	rollback_semantics
		if the operation temporarily cannot be performed
Violations:	`ProcessMappingViolation`	In case InstanceSpecifier does not point to a PortPrototype typed by a CryptoCertificateGroupInterface modeled for the current process.
	`InsufficientPermissionsViolation`	In case the current process does not have write permissions, i.e., CryptoCertificateGroupToPortPrototypeMapping.writeAccess != True
Description:	Remove a certificate from a certificate group.	

⌋

8.37.1.2.3 Named Constructors

8.37.1.2.3.1 LoadCertificate

[SWS_CRYPT_40641] Definition of API function ara::crypto::x509::X509Provider::LoadCertificate

Upstream requirements: RS_CRYPTO_02306

⌈

Kind:	function	
Header file:	#include "ara/crypto/x509/x509_provider.h"	
Scope:	`class ara::crypto::x509::X509Provider`	
Syntax:	`virtual ara::core::Result< Certificate::Sptrc > LoadCertificate (const ara::core::InstanceSpecifier &instanceSpecifier) noexcept=0;`	
Parameters (in):	instanceSpecifier	An `ara::core::InstanceSpecifier` identifying a storage location of a certificate.
Return value:	ara::core::Result< Certificate::Sptrc >	an Smart pointer to the instantiated certificate
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::crypto::CryptoErrc::kEmptyCertificateSlot	rollback_semantics
		The CryptoCertificate identified by the provided InstanceSpecifier does not contain a persisted Certificate
Violations:	`ProcessMappingViolation`	In case InstanceSpecifier does not point to a PortPrototype typed by a CryptoCertificateInterface modeled for the current process.
Description:	Load a certificate from the persistent certificate storage.	

⌋

8.37.1.2.3.2 LoadCertificateGroup

[SWS_CRYPT_50126] Definition of API function ara::crypto::x509::X509Provider::LoadCertificateGroup

Upstream requirements: RS_CRYPTO_02306

⌈

Kind:	function	
Header file:	#include "ara/crypto/x509/x509_provider.h"	
Scope:	`class ara::crypto::x509::X509Provider`	
Syntax:	`virtual ara::core::Result< ara::core::Vector< Certificate::Sptrc > > LoadCertificateGroup (const ara::core::InstanceSpecifier &instanceSpecifier) noexcept=0;`	
Parameters (in):	instanceSpecifier	An instance specifier identifying a storage location of a certificate group.
Return value:	ara::core::Result< ara::core::Vector< Certificate::Sptrc > >	a vector containing the certificates stored in the certificate group.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	This function does not specify any standardized errors.	
Violations:	`ProcessMappingViolation`	In case InstanceSpecifier does not point to a PortPrototype typed by a CryptoCertificateGroupInterface modeled for the current process.
Description:	Load a group of certificates from the persistent certificate storage.	

⌋

9 Service Interfaces

This functional cluster does not define any provided or required service interfaces.

10 Configuration

The configuration model of this functional cluster is defined in [16]. This chapter defines the default values for attributes and semantic constraints for elements specified in [16] that are part of the configuration model of this functional cluster.

10.1 Default Values

This functional cluster does not define any default values for attributes specified in [16].

10.2 Semantic Constraints

This section defines semantic constraints for elements specified in [16] that are part of the configuration model of this functional cluster.

[SWS_CRYPT_CONSTR_00001] Configurable Namespace for Cryptography ⌈ `CryptoInterface.namespace` shall never exist.⌋

A Mentioned Manifest Elements

This chapter contains the remaining set of meta-class tables which are not shown directly in the main body of this document.

This chapter is generated.

Class	***AbstractCryptoKeySlotToPortPrototypeMapping*** (abstract)			
Note	This abstract meta-class acts as a base class for crypto key slot mappings. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDeploymentElement*, *UploadablePackageElement*			
Subclasses	CryptoKeySlotToClientPortPrototypeMapping, CryptoKeySlotToPortPrototypeMapping			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
allowPlainText Export	Boolean	0..1	attr	This attribute describes whether the key slot content is allowed to be exported in plaintext format or not by the application referenced in the role process. Plaintext export is only possible if supported by the cryptographic module and by the CryptoProvider implementation and if the key has been generated/ imported with the keySlot.keySlotContentAllowed Usage.allowedKeyslotUsage set to exportable.
allowPlainText Import	Boolean	0..1	attr	This attribute describes whether the key slot content is allowed to be imported in plaintext format or not by the application referenced in the role process. Plaintext import is only possible if supported by the cryptographic module and by the CryptoProvider implementation.
keySlot	CryptoKeySlot	0..1	ref	This reference represents the mapped CryptoKeySlot.
portPrototype	RPortPrototype	0..1	iref	This reference represents the mapped PortPrototype. **InstanceRef implemented by:** RPortPrototypeIn ExecutableInstanceRef
process	Process	0..1	ref	This reference represents the process required as context for the mapping.

Table A.1: AbstractCryptoKeySlotToPortPrototypeMapping

Class	**ArtifactLocator**			
Note	This meta-class has the ability to define the location of an artifact that is represented by a model element, e.g. Executable. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	SoftwareCluster.artifactLocator, SoftwarePackage.artifactLocator			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
represented ModelElement	Identifiable	0..1	ref	This reference identifies the model element that is represented by the artifact.
uri	String	0..1	attr	This attribute describes the location of the artifact.

Table A.2: ArtifactLocator

Class	**ComSecOcToCryptoKeySlotMapping**			
Note	This meta-class maps the ServiceElementSecureComConfig defined in the COM deployment to the CryptoKeySlot defined in the Crypto Stack. **Tags:** atp.recommendedPackage=FCInteractions This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *FunctionalClusterInteractsWithFunctionalClusterMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadableDeployment Element*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
cryptoKeySlot	CryptoKeySlot	0..1	ref	Reference to the CryptoKeySlot in the Crypto Stack.
serviceElement SecureCom Config	ServiceInterface ElementSecureCom Config	0..1	ref	Reference to the ServiceInterfaceElementSecureCom Config element in the COM config.

Table A.3: ComSecOcToCryptoKeySlotMapping

Class	**CryptoCertificate**			
Note	This meta-class represents the ability to model a cryptographic certificate. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	CryptoModuleInstantiation.cryptoCertificate			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
isPrivate	Boolean	0..1	attr	This attribute controls the possibility to access the content of the CryptoCertificateSlot by Find() interfaces of the X509 Provider.

Table A.4: CryptoCertificate

Class	**CryptoCertificateGroup**			
Note	This meta-class represents the ability to model a group of cryptographic certificates. **Tags:** atp.Status=candidate This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	CryptoModuleInstantiation.cryptoCertificateGroup			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
capacity	PositiveInteger	0..1	attr	Capacity of the certificate group in bytes of DER serialized certificates. **Tags:** atp.Status=candidate
included Certificate	CryptoCertificate	*	ref	This optional collection of references to individual Crypto CErtificates can be used to include some Crypto Certificates "by identity" into the CryptoCertificateGroup. **Tags:** atp.Status=candidate
label	String	0..1	attr	Label for application-level identification of certificate group. **Tags:** atp.Status=candidate

Table A.5: CryptoCertificateGroup

Class	CryptoCertificateGroupInterface			
Note	This meta-class provides the ability to define a PortInterface for a group of CryptoCertificates. **Tags:** atp.Status=candidate atp.recommendedPackage=CryptoInterfaces This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *CryptoInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
capacity	PositiveInteger	0..1	attr	Capacity of the certificate group in bytes of DER serialized certificates. **Tags:** atp.Status=candidate

Table A.6: CryptoCertificateGroupInterface

Class	CryptoCertificateGroupToPortPrototypeMapping			
Note	This meta-class represents the ability to define a mapping between a CryptoCertificateGroup on target-configuration level to a given PortPrototype that is typed by a CryptoCertificateGroupInterface. **Tags:** atp.Status=candidate atp.recommendedPackage=CryptoMappipngs This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDeploymentElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
certificateGroup	CryptoCertificateGroup	0..1	ref	This reference identifies the refernce group mapped to the PortPrototype. **Tags:** atp.Status=candidate
portPrototype	RPortPrototype	0..1	iref	This reference represents the mapped PortPrototype. **Stereotypes:** atpUriDef **Tags:** atp.Status=candidate **InstanceRef implemented by:** RPortPrototypeIn ExecutableInstanceRef
process	Process	0..1	ref	This reference represents the process required as context for the mapping.This reference represents the process required as context for the mapping. **Tags:** atp.Status=candidate

Table A.7: CryptoCertificateGroupToPortPrototypeMapping

Class	CryptoCertificateInterface			
Note	This meta-class provides the ability to define a PortInterface for a CryptoCertificate. **Tags:** atp.Status=candidate atp.recommendedPackage=CryptoInterfaces This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *CryptoInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
isPrivate	Boolean	0..1	attr	This attribute controls the possibility to access the content of the CryptoCertificateSlot by Find() interfaces of the X509 Provider. **Tags:** atp.Status=candidate

▽

△

Class	**CryptoCertificateInterface**			
writeAccess	Boolean	0..1	attr	This attribute defines whether the application has write-access to the CryptoCertificate (true) or only read-access (false). **Tags:** atp.Status=candidate

Table A.8: CryptoCertificateInterface

Class	**CryptoCertificateToPortPrototypeMapping**			
Note	This meta-class represents the ability to define a mapping between a CryptoCertificate on target-configuration level to a given PortPrototype that is typed by a CryptoCertificateInterface. **Tags:** atp.recommendedPackage=CryptoCertificateToPortPrototypeMappings This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDeploymentElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
crypto Certificate	CryptoCertificate	0..1	ref	This reference represents the mapped cryptoCertificate.
portPrototype	RPortPrototype	0..1	iref	This reference represents the mapped PortPrototype. **InstanceRef implemented by:** RPortPrototypeIn ExecutableInstanceRef
process	Process	0..1	ref	This reference represents the process required as context for the mapping.
writeAccess	Boolean	0..1	attr	This attribute defines whether the application has write-access to the CryptoCertificate (true) or only read-access (false).

Table A.9: CryptoCertificateToPortPrototypeMapping

Class	***CryptoInterface*** (abstract)			
Note	This meta-class provides the abstract ability to define a PortInterface for the support of crypto use cases. **Tags:** atp.Status=candidate This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Subclasses	*AbstractCryptoKeySlotInterface*, CryptoCertificateGroupInterface, CryptoCertificateInterface, Crypto ProviderInterface			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table A.10: CryptoInterface

Class	**CryptoKeySlot**			
Note	This meta-class represents the ability to define a concrete key to be used for a crypto operation. **Tags:** atp.ManifestKind=MachineManifest This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	CryptoProvider.keySlot			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

▽

△

Class	**CryptoKeySlot**			
algorithm Description	CryptoAlgorithm Description	*	aggr	This aggregation contains the collection of crypto algorithm descriptions that can be used in the context of the enclosing crypto key slot. **Tags:** atp.Status=candidate
allocateShadow Copy	Boolean	0..1	attr	This attribute defines whether a shadow copy of this Key Slot shall be allocated to enable rollback of a failed Key Slot update campaign (see interface BeginTransaction).
cryptoAlgId	String	0..1	attr	This attribute defines a crypto algorithm restriction (kAlgId Any means without restriction). The algorithm can be specified partially: family & length, mode, padding. Future Crypto Providers can support some crypto algorithms that are not well known/ standardized today, therefore AUTOSAR doesn't provide a concrete list of crypto algorithms' identifiers and doesn't suppose usage of numerical identifiers. Instead of this a provider supplier should provide string names of supported algorithms in accompanying documentation. The name of a crypto algorithm shall follow the rules defined in the specification of cryptography for Adaptive Platform.
cryptoKeySlot Design	CryptoKeySlotDesign	0..1	ref	This reference identifies the CryptoKeySlotDesign from which the referencing CryptoKeySlot was derived.
cryptoObject Type	CryptoObjectTypeEnum	0..1	attr	Object type that can be stored in the slot. If this field contains "Undefined" then mSlotCapacity must be provided and larger then 0. **Tags:** atp.Status=candidate
keySlotAllowed Modification	CryptoKeySlotAllowed Modification	0..1	aggr	Restricts how this keySlot may be used **Tags:** atp.Status=candidate
keySlotContent AllowedUsage	CryptoKeySlotContent AllowedUsage	*	aggr	Restriction of allowed usage of a key stored to the slot. **Tags:** atp.Status=candidate
slotCapacity	PositiveInteger	0..1	attr	Capacity of the slot in bytes to be reserved by the stack vendor. One use case is to define this value in case that the cryptoObjectType is undefined and the slot size can not be deduced from cryptoObjectType and cryptoAlgId. "0" means slot size can be deduced from cryptoObject Type and cryptoAlgId.
slotType	CryptoKeySlotType Enum	0..1	attr	This attribute defines whether the keySlot is exclusively used by the Application; or whether it is used by Stack Services and managed by a Key Manager Application. **Tags:** atp.Status=candidate

Table A.11: CryptoKeySlot

Class	**CryptoKeySlotClientInterface**			
Note	This meta-class provides the ability to define a PortInterface restricted to **using** Crypto Key Slots. **Tags:** atp.Status=candidate atp.recommendedPackage=CryptoInterfaces This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *AbstractCryptoKeySlotInterface*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *CryptoInterface*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table A.12: CryptoKeySlotClientInterface

Class	**CryptoKeySlotInterface**			
Note	This meta-class provides the ability to define a PortInterface for **using and modifying** Crypto Key Slots. **Tags:** atp.Status=candidate atp.recommendedPackage=CryptoInterfaces This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *AbstractCryptoKeySlotInterface*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *CryptoInterface*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table A.13: CryptoKeySlotInterface

Class	**CryptoKeySlotToClientPortPrototypeMapping**			
Note	This meta-class represents the ability to define a mapping between a CryptoKeySlot on deployment level to a given PortPrototype that is typed by a CryptoKeyClientSlotInterface (this means only read is supported). **Tags:** atp.recommendedPackage=CryptoKeySlotMappings This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *AbstractCryptoKeySlotToPortPrototypeMapping*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadableDeployment Element*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table A.14: CryptoKeySlotToClientPortPrototypeMapping

Class	**CryptoKeySlotToPortPrototypeMapping**			
Note	This meta-class represents the ability to define a mapping between a CryptoKeySlot on target-configuration level to a given PortPrototype that is typed by a CryptoKeySlotInterface. **Tags:** atp.recommendedPackage=CryptoKeySlotMappings This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *AbstractCryptoKeySlotToPortPrototypeMapping*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadableDeployment Element*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table A.15: CryptoKeySlotToPortPrototypeMapping

Class	**CryptoProviderInterface**			
Note	This meta-class provides the ability to define a PortInterface for a CryptoProvider. **Tags:** atp.Status=candidate atp.recommendedPackage=CryptoInterfaces This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *CryptoInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table A.16: CryptoProviderInterface

Class	***PortInterface*** (abstract)			
Note	Abstract base class for an interface that is either provided or required by a port of a software component.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Subclasses	*AbstractRawDataStreamInterface*, *AbstractSuspendToRamInterface*, *AbstractSynchronizedTimeBase Interface*, ClientServerInterface, *CryptoInterface*, *DataInterface*, *DiagnosticPortInterface*, FirewallState SwitchInterface, *IdsmAbstractPortInterface*, LogAndTraceInterface, ModeSwitchInterface, Network ManagementPortInterface, *PersistencyInterface*, *PlatformHealthManagementInterface*, ServiceInterface, *StateManagementPortInterface*, TriggerInterface			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
namespace (ordered)	SymbolProps	*	aggr	This represents the SymbolProps used for the definition of a hierarchical namespace applicable for the generation of code artifacts out of the definition of a ServiceInterface. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=namespace.shortName This Attribute is only used by the AUTOSAR Adaptive Platform.

Table A.17: PortInterface

Class	**Process**			
Note	This meta-class provides information required to execute the referenced `Executable`. **Tags:** atp.recommendedPackage=Processes This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *AbstractExecutionContext*, *AtpClassifier*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadableDeploymentElement*, *Uploadable PackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
design	ProcessDesign	0..1	ref	This reference represents the identification of the design-time representation for the Process that owns the reference.
executable	Executable	*	ref	Reference to executable that is executed in the process. **Stereotypes:** atpUriDef
functionCluster Affiliation	String	0..1	attr	This attribute specifies which functional cluster the Process is affiliated with.
numberOf RestartAttempts	PositiveInteger	0..1	attr	This attribute defines how often a process shall be restarted if the start fails. numberOfRestartAttempts = "0" OR Attribute not existing, start once numberOfRestartAttempts = "1", start a second time
preMapping	Boolean	0..1	attr	This attribute describes whether the executable is preloaded into the memory.
processState Machine	ModeDeclarationGroup Prototype	0..1	aggr	Set of Process States that are defined for the process. This attribute is used to support the modeling of execution dependencies that utilize the condition of process state. Please note that the process states may not be modeled arbitrarily at any stage of the AUTOSAR workflow because the supported states are standardized in the context of the SWS Execution Management [17].
stateDependent StartupConfig	StateDependentStartup Config	*	aggr	Applicable startup configurations.

Table A.18: Process

Class	**RPortPrototype**			
Note	Component port requiring a certain port interface.			
Base	*ARObject*, *AbstractRequiredPortPrototype*, *AtpBlueprintable*, *AtpFeature*, *AtpPrototype*, *Identifiable*, *MultilanguageReferrable*, *PortPrototype*, *Referrable*			
Aggregated by	*AtpClassifier*.atpFeature, *SwComponentType*.port			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
required Interface	PortInterface	0..1	tref	The interface that this port requires. **Stereotypes:** isOfType

Table A.19: RPortPrototype

B Interfaces to other Functional Clusters (informative)

B.1 Overview

AUTOSAR decided not to standardize interfaces which are exclusively used between Functional Clusters (on platform-level only), to allow efficient implementations, which might depend e.g. on the used Operating System.

This chapter provides informative guidelines how the interaction between Functional Clusters looks like, by clustering the relevant requirements of this document to describe Inter-Functional Cluster (IFC) interfaces. In addition, the standardized public interfaces which are accessible by user space applications (see chapters 8 and 9) can also be used for interaction between Functional Clusters.

The goal is to provide a clear understanding of Functional Cluster boundaries and interaction, without specifying syntactical details. This ensures compatibility between documents specifying different Functional Clusters and supports parallel implementation of different Functional Clusters. Details of the interfaces are up to the platform provider. Additional interfaces, parameters and return values can be added.

B.2 Interface Tables

No content defined.

C Change History

This chapter provides an overview of the history of constraints and specification items. Please note that the lists in this chapter also include constraints and specification items that have been removed from the specification in a later version. These constraints and specification items do not appear as hyperlinks in the document.

C.1 Change History of this document according to AUTOSAR Release R22-11

C.1.1 Added Specification Items in R22-11

[SWS_CRYPT_00004] [SWS_CRYPT_00005] [SWS_CRYPT_00006] [SWS_CRYPT_00007] [SWS_CRYPT_00009] [SWS_CRYPT_00101] [SWS_CRYPT_00102] [SWS_CRYPT_00103] [SWS_CRYPT_00104] [SWS_CRYPT_00500] [SWS_CRYPT_00501] [SWS_CRYPT_00502] [SWS_CRYPT_00503] [SWS_CRYPT_00504] [SWS_CRYPT_00505] [SWS_CRYPT_00506] [SWS_CRYPT_00507] [SWS_CRYPT_00508] [SWS_CRYPT_00601] [SWS_CRYPT_00603] [SWS_CRYPT_00608] [SWS_CRYPT_00609] [SWS_CRYPT_00610] [SWS_CRYPT_00611] [SWS_CRYPT_00622] [SWS_CRYPT_00901] [SWS_CRYPT_00902] [SWS_CRYPT_00903] [SWS_CRYPT_00905] [SWS_CRYPT_00906] [SWS_CRYPT_00907] [SWS_CRYPT_00908] [SWS_CRYPT_00909] [SWS_CRYPT_00910] [SWS_CRYPT_00919] [SWS_CRYPT_01200] [SWS_CRYPT_01201] [SWS_CRYPT_01202] [SWS_CRYPT_01203] [SWS_CRYPT_01204] [SWS_CRYPT_01207] [SWS_CRYPT_01208] [SWS_CRYPT_01209] [SWS_CRYPT_01210] [SWS_CRYPT_01211] [SWS_CRYPT_01213] [SWS_CRYPT_01501] [SWS_CRYPT_01502] [SWS_CRYPT_01503] [SWS_CRYPT_01504] [SWS_CRYPT_01506] [SWS_CRYPT_01508] [SWS_CRYPT_01651] [SWS_CRYPT_01653] [SWS_CRYPT_01654] [SWS_CRYPT_01655] [SWS_CRYPT_01656] [SWS_CRYPT_01657] [SWS_CRYPT_01658] [SWS_CRYPT_01659] [SWS_CRYPT_01660] [SWS_CRYPT_01661] [SWS_CRYPT_01662] [SWS_CRYPT_01800] [SWS_CRYPT_01801] [SWS_CRYPT_01802] [SWS_CRYPT_01803] [SWS_CRYPT_01804] [SWS_CRYPT_01805] [SWS_CRYPT_01806] [SWS_CRYPT_01807] [SWS_CRYPT_01808] [SWS_CRYPT_01811] [SWS_CRYPT_01820] [SWS_CRYPT_01821] [SWS_CRYPT_01822] [SWS_CRYPT_01823] [SWS_CRYPT_02104] [SWS_CRYPT_02105] [SWS_CRYPT_02106] [SWS_CRYPT_02107] [SWS_CRYPT_02108] [SWS_CRYPT_02109] [SWS_CRYPT_02121] [SWS_CRYPT_02122] [SWS_CRYPT_02123] [SWS_CRYPT_02400] [SWS_CRYPT_02408] [SWS_CRYPT_02409] [SWS_CRYPT_02410] [SWS_CRYPT_02411] [SWS_CRYPT_02412] [SWS_CRYPT_02413] [SWS_CRYPT_02414] [SWS_CRYPT_02415] [SWS_CRYPT_02416] [SWS_CRYPT_02417] [SWS_CRYPT_02418] [SWS_CRYPT_02419] [SWS_CRYPT_02420] [SWS_CRYPT_02421] [SWS_CRYPT_02422] [SWS_CRYPT_02700] [SWS_CRYPT_02701] [SWS_CRYPT_02702] [SWS_CRYPT_02703] [SWS_CRYPT_02704] [SWS_CRYPT_02705] [SWS_CRYPT_02706] [SWS_CRYPT_02726] [SWS_CRYPT_03000] [SWS_CRYPT_

03002] [SWS_CRYPT_03003] [SWS_CRYPT_03004] [SWS_CRYPT_03005] [SWS_CRYPT_03006] [SWS_CRYPT_03007] [SWS_CRYPT_03008] [SWS_CRYPT_03009] [SWS_CRYPT_03300] [SWS_CRYPT_03301] [SWS_CRYPT_03302] [SWS_CRYPT_03303] [SWS_CRYPT_03304] [SWS_CRYPT_03305] [SWS_CRYPT_03311] [SWS_CRYPT_03312] [SWS_CRYPT_03313] [SWS_CRYPT_03904] [SWS_CRYPT_03905] [SWS_CRYPT_03906] [SWS_CRYPT_03910] [SWS_CRYPT_04200] [SWS_CRYPT_04202] [SWS_CRYPT_04203] [SWS_CRYPT_04204] [SWS_CRYPT_04205] [SWS_CRYPT_04207] [SWS_CRYPT_04208] [SWS_CRYPT_04209] [SWS_CRYPT_04213] [SWS_CRYPT_10000] [SWS_CRYPT_10003] [SWS_CRYPT_10004] [SWS_CRYPT_10005] [SWS_CRYPT_10014] [SWS_CRYPT_10015] [SWS_CRYPT_10016] [SWS_CRYPT_10017] [SWS_CRYPT_10018] [SWS_CRYPT_10019] [SWS_CRYPT_10031] [SWS_CRYPT_10033] [SWS_CRYPT_10042] [SWS_CRYPT_10099] [SWS_CRYPT_10100] [SWS_CRYPT_10101] [SWS_CRYPT_10102] [SWS_CRYPT_10111] [SWS_CRYPT_10112] [SWS_CRYPT_10113] [SWS_CRYPT_10114] [SWS_CRYPT_10115] [SWS_CRYPT_10150] [SWS_CRYPT_10151] [SWS_CRYPT_10152] [SWS_CRYPT_10153] [SWS_CRYPT_10154] [SWS_CRYPT_10155] [SWS_CRYPT_10200] [SWS_CRYPT_10300] [SWS_CRYPT_10301] [SWS_CRYPT_10303] [SWS_CRYPT_10304] [SWS_CRYPT_10305] [SWS_CRYPT_10306] [SWS_CRYPT_10400] [SWS_CRYPT_10401] [SWS_CRYPT_10402] [SWS_CRYPT_10403] [SWS_CRYPT_10411] [SWS_CRYPT_10412] [SWS_CRYPT_10413] [SWS_CRYPT_10451] [SWS_CRYPT_10452] [SWS_CRYPT_10453] [SWS_CRYPT_10454] [SWS_CRYPT_10455] [SWS_CRYPT_10456] [SWS_CRYPT_10700] [SWS_CRYPT_10701] [SWS_CRYPT_10710] [SWS_CRYPT_10711] [SWS_CRYPT_10712] [SWS_CRYPT_10750] [SWS_CRYPT_10751] [SWS_CRYPT_10752] [SWS_CRYPT_10753] [SWS_CRYPT_10800] [SWS_CRYPT_10801] [SWS_CRYPT_10802] [SWS_CRYPT_10808] [SWS_CRYPT_10810] [SWS_CRYPT_10811] [SWS_CRYPT_10812] [SWS_CRYPT_10813] [SWS_CRYPT_10814] [SWS_CRYPT_10815] [SWS_CRYPT_10816] [SWS_CRYPT_10817] [SWS_CRYPT_10818] [SWS_CRYPT_10819] [SWS_CRYPT_10821] [SWS_CRYPT_10822] [SWS_CRYPT_10823] [SWS_CRYPT_10850] [SWS_CRYPT_10851] [SWS_CRYPT_10852] [SWS_CRYPT_10853] [SWS_CRYPT_13000] [SWS_CRYPT_13001] [SWS_CRYPT_13002] [SWS_CRYPT_13003] [SWS_CRYPT_13100] [SWS_CRYPT_13101] [SWS_CRYPT_13102] [SWS_CRYPT_13103] [SWS_CRYPT_13104] [SWS_CRYPT_13105] [SWS_CRYPT_13106] [SWS_CRYPT_13107] [SWS_CRYPT_13108] [SWS_CRYPT_13109] [SWS_CRYPT_13110] [SWS_CRYPT_13111] [SWS_CRYPT_13112] [SWS_CRYPT_13113] [SWS_CRYPT_13114] [SWS_CRYPT_13115] [SWS_CRYPT_13116] [SWS_CRYPT_13117] [SWS_CRYPT_13118] [SWS_CRYPT_13119] [SWS_CRYPT_13120] [SWS_CRYPT_13121] [SWS_CRYPT_13122] [SWS_CRYPT_19900] [SWS_CRYPT_19902] [SWS_CRYPT_19903] [SWS_CRYPT_19904] [SWS_CRYPT_19905] [SWS_CRYPT_19906] [SWS_CRYPT_19950] [SWS_CRYPT_19951] [SWS_CRYPT_19953] [SWS_CRYPT_19954] [SWS_CRYPT_20000] [SWS_CRYPT_20001] [SWS_CRYPT_20002] [SWS_CRYPT_20003] [SWS_CRYPT_20004] [SWS_CRYPT_20005] [SWS_CRYPT_20006] [SWS_CRYPT_20007] [SWS_CRYPT_20009] [SWS_CRYPT_20010] [SWS_CRYPT_20011] [SWS_CRYPT_20098] [SWS_CRYPT_20099] [SWS_CRYPT_20100] [SWS_CRYPT_20101] [SWS_CRYPT_20102] [SWS_CRYPT_

20103] [SWS_CRYPT_20301] [SWS_CRYPT_20302] [SWS_CRYPT_20303] [SWS_CRYPT_20304] [SWS_CRYPT_20312] [SWS_CRYPT_20313] [SWS_CRYPT_20314] [SWS_CRYPT_20316] [SWS_CRYPT_20319] [SWS_CRYPT_20400] [SWS_CRYPT_20401] [SWS_CRYPT_20402] [SWS_CRYPT_20411] [SWS_CRYPT_20412] [SWS_CRYPT_20414] [SWS_CRYPT_20500] [SWS_CRYPT_20501] [SWS_CRYPT_20502] [SWS_CRYPT_20503] [SWS_CRYPT_20504] [SWS_CRYPT_20505] [SWS_CRYPT_20506] [SWS_CRYPT_20507] [SWS_CRYPT_20512] [SWS_CRYPT_20513] [SWS_CRYPT_20514] [SWS_CRYPT_20515] [SWS_CRYPT_20516] [SWS_CRYPT_20517] [SWS_CRYPT_20518] [SWS_CRYPT_20600] [SWS_CRYPT_20601] [SWS_CRYPT_20602] [SWS_CRYPT_20603] [SWS_CRYPT_20611] [SWS_CRYPT_20612] [SWS_CRYPT_20613] [SWS_CRYPT_20614] [SWS_CRYPT_20615] [SWS_CRYPT_20616] [SWS_CRYPT_20617] [SWS_CRYPT_20618] [SWS_CRYPT_20619] [SWS_CRYPT_20641] [SWS_CRYPT_20643] [SWS_CRYPT_20644] [SWS_CRYPT_20651] [SWS_CRYPT_20652] [SWS_CRYPT_20654] [SWS_CRYPT_20700] [SWS_CRYPT_20701] [SWS_CRYPT_20703] [SWS_CRYPT_20710] [SWS_CRYPT_20711] [SWS_CRYPT_20712] [SWS_CRYPT_20721] [SWS_CRYPT_20722] [SWS_CRYPT_20723] [SWS_CRYPT_20724] [SWS_CRYPT_20725] [SWS_CRYPT_20726] [SWS_CRYPT_20727] [SWS_CRYPT_20728] [SWS_CRYPT_20729] [SWS_CRYPT_20730] [SWS_CRYPT_20731] [SWS_CRYPT_20732] [SWS_CRYPT_20733] [SWS_CRYPT_20741] [SWS_CRYPT_20742] [SWS_CRYPT_20743] [SWS_CRYPT_20744] [SWS_CRYPT_20745] [SWS_CRYPT_20746] [SWS_CRYPT_20747] [SWS_CRYPT_20748] [SWS_CRYPT_20750] [SWS_CRYPT_20751] [SWS_CRYPT_20752] [SWS_CRYPT_20753] [SWS_CRYPT_20754] [SWS_CRYPT_20755] [SWS_CRYPT_20756] [SWS_CRYPT_20757] [SWS_CRYPT_20758] [SWS_CRYPT_20760] [SWS_CRYPT_20761] [SWS_CRYPT_20762] [SWS_CRYPT_20763] [SWS_CRYPT_20764] [SWS_CRYPT_20765] [SWS_CRYPT_20800] [SWS_CRYPT_20801] [SWS_CRYPT_20802] [SWS_CRYPT_20810] [SWS_CRYPT_20811] [SWS_CRYPT_20812] [SWS_CRYPT_20813] [SWS_CRYPT_20901] [SWS_CRYPT_20902] [SWS_CRYPT_20903] [SWS_CRYPT_20904] [SWS_CRYPT_20905] [SWS_CRYPT_20906] [SWS_CRYPT_20907] [SWS_CRYPT_20908] [SWS_CRYPT_20909] [SWS_CRYPT_20910] [SWS_CRYPT_21000] [SWS_CRYPT_21001] [SWS_CRYPT_21002] [SWS_CRYPT_21010] [SWS_CRYPT_21011] [SWS_CRYPT_21012] [SWS_CRYPT_21013] [SWS_CRYPT_21100] [SWS_CRYPT_21101] [SWS_CRYPT_21102] [SWS_CRYPT_21110] [SWS_CRYPT_21111] [SWS_CRYPT_21112] [SWS_CRYPT_21113] [SWS_CRYPT_21114] [SWS_CRYPT_21115] [SWS_CRYPT_21116] [SWS_CRYPT_21117] [SWS_CRYPT_21118] [SWS_CRYPT_21300] [SWS_CRYPT_21301] [SWS_CRYPT_21302] [SWS_CRYPT_21311] [SWS_CRYPT_21312] [SWS_CRYPT_21313] [SWS_CRYPT_21314] [SWS_CRYPT_21315] [SWS_CRYPT_21400] [SWS_CRYPT_21401] [SWS_CRYPT_21402] [SWS_CRYPT_21411] [SWS_CRYPT_21412] [SWS_CRYPT_21413] [SWS_CRYPT_21414] [SWS_CRYPT_21415] [SWS_CRYPT_21416] [SWS_CRYPT_21500] [SWS_CRYPT_21501] [SWS_CRYPT_21512] [SWS_CRYPT_21513] [SWS_CRYPT_21514] [SWS_CRYPT_21515] [SWS_CRYPT_21516] [SWS_CRYPT_21517] [SWS_CRYPT_21519] [SWS_CRYPT_21520] [SWS_CRYPT_21521] [SWS_CRYPT_21522] [SWS_CRYPT_21523] [SWS_

CRYPT_21524] [SWS_CRYPT_21525] [SWS_CRYPT_21715] [SWS_CRYPT_21800] [SWS_CRYPT_21801] [SWS_CRYPT_21802] [SWS_CRYPT_21810] [SWS_CRYPT_21813] [SWS_CRYPT_21815] [SWS_CRYPT_21816] [SWS_CRYPT_21817] [SWS_CRYPT_21818] [SWS_CRYPT_22100] [SWS_CRYPT_22101] [SWS_CRYPT_22102] [SWS_CRYPT_22110] [SWS_CRYPT_22111] [SWS_CRYPT_22112] [SWS_CRYPT_22113] [SWS_CRYPT_22114] [SWS_CRYPT_22115] [SWS_CRYPT_22116] [SWS_CRYPT_22117] [SWS_CRYPT_22118] [SWS_CRYPT_22119] [SWS_CRYPT_22120] [SWS_CRYPT_22200] [SWS_CRYPT_22201] [SWS_CRYPT_22210] [SWS_CRYPT_22211] [SWS_CRYPT_22212] [SWS_CRYPT_22213] [SWS_CRYPT_22214] [SWS_CRYPT_22215] [SWS_CRYPT_22216] [SWS_CRYPT_22500] [SWS_CRYPT_22501] [SWS_CRYPT_22503] [SWS_CRYPT_22511] [SWS_CRYPT_22700] [SWS_CRYPT_22701] [SWS_CRYPT_22702] [SWS_CRYPT_22711] [SWS_CRYPT_22712] [SWS_CRYPT_22713] [SWS_CRYPT_22900] [SWS_CRYPT_22901] [SWS_CRYPT_22902] [SWS_CRYPT_22911] [SWS_CRYPT_22912] [SWS_CRYPT_22913] [SWS_CRYPT_22914] [SWS_CRYPT_22915] [SWS_CRYPT_23000] [SWS_CRYPT_23001] [SWS_CRYPT_23002] [SWS_CRYPT_23003] [SWS_CRYPT_23011] [SWS_CRYPT_23012] [SWS_CRYPT_23013] [SWS_CRYPT_23014] [SWS_CRYPT_23015] [SWS_CRYPT_23016] [SWS_CRYPT_23200] [SWS_CRYPT_23201] [SWS_CRYPT_23210] [SWS_CRYPT_23211] [SWS_CRYPT_23212] [SWS_CRYPT_23213] [SWS_CRYPT_23214] [SWS_CRYPT_23215] [SWS_CRYPT_23216] [SWS_CRYPT_23300] [SWS_CRYPT_23301] [SWS_CRYPT_23302] [SWS_CRYPT_23311] [SWS_CRYPT_23312] [SWS_CRYPT_23500] [SWS_CRYPT_23501] [SWS_CRYPT_23510] [SWS_CRYPT_23511] [SWS_CRYPT_23512] [SWS_CRYPT_23513] [SWS_CRYPT_23514] [SWS_CRYPT_23515] [SWS_CRYPT_23516] [SWS_CRYPT_23600] [SWS_CRYPT_23601] [SWS_CRYPT_23602] [SWS_CRYPT_23611] [SWS_CRYPT_23612] [SWS_CRYPT_23613] [SWS_CRYPT_23614] [SWS_CRYPT_23615] [SWS_CRYPT_23616] [SWS_CRYPT_23617] [SWS_CRYPT_23618] [SWS_CRYPT_23619] [SWS_CRYPT_23620] [SWS_CRYPT_23621] [SWS_CRYPT_23622] [SWS_CRYPT_23623] [SWS_CRYPT_23624] [SWS_CRYPT_23625] [SWS_CRYPT_23626] [SWS_CRYPT_23627] [SWS_CRYPT_23634] [SWS_CRYPT_23635] [SWS_CRYPT_23700] [SWS_CRYPT_23701] [SWS_CRYPT_23702] [SWS_CRYPT_23710] [SWS_CRYPT_23711] [SWS_CRYPT_23712] [SWS_CRYPT_23715] [SWS_CRYPT_23716] [SWS_CRYPT_23717] [SWS_CRYPT_23800] [SWS_CRYPT_23801] [SWS_CRYPT_23802] [SWS_CRYPT_23911] [SWS_CRYPT_24000] [SWS_CRYPT_24001] [SWS_CRYPT_24002] [SWS_CRYPT_24011] [SWS_CRYPT_24012] [SWS_CRYPT_24013] [SWS_CRYPT_24014] [SWS_CRYPT_24015] [SWS_CRYPT_24016] [SWS_CRYPT_24017] [SWS_CRYPT_24018] [SWS_CRYPT_24019] [SWS_CRYPT_24100] [SWS_CRYPT_24101] [SWS_CRYPT_24102] [SWS_CRYPT_24111] [SWS_CRYPT_24112] [SWS_CRYPT_24113] [SWS_CRYPT_24114] [SWS_CRYPT_24115] [SWS_CRYPT_24116] [SWS_CRYPT_24400] [SWS_CRYPT_24401] [SWS_CRYPT_24410] [SWS_CRYPT_24411] [SWS_CRYPT_24412] [SWS_CRYPT_24413] [SWS_CRYPT_24414] [SWS_CRYPT_24415] [SWS_CRYPT_24714] [SWS_CRYPT_24715] [SWS_CRYPT_24800] [SWS_CRYPT_24801] [SWS_CRYPT_24802] [SWS_CRYPT_24811] [SWS_CRYPT_29000] [SWS_CRYPT_29001] [SWS_CRYPT_29002] [SWS_CRYPT_29003] [SWS_CRYPT_29004] [SWS_

CRYPT_29010] [SWS_CRYPT_29011] [SWS_CRYPT_29012] [SWS_CRYPT_29013] [SWS_CRYPT_29014] [SWS_CRYPT_29015] [SWS_CRYPT_29020] [SWS_CRYPT_29021] [SWS_CRYPT_29022] [SWS_CRYPT_29023] [SWS_CRYPT_29024] [SWS_CRYPT_29030] [SWS_CRYPT_29031] [SWS_CRYPT_29032] [SWS_CRYPT_29033] [SWS_CRYPT_29034] [SWS_CRYPT_29035] [SWS_CRYPT_29040] [SWS_CRYPT_29041] [SWS_CRYPT_29042] [SWS_CRYPT_29043] [SWS_CRYPT_29044] [SWS_CRYPT_29045] [SWS_CRYPT_29046] [SWS_CRYPT_29047] [SWS_CRYPT_29048] [SWS_CRYPT_29049] [SWS_CRYPT_30001] [SWS_CRYPT_30002] [SWS_CRYPT_30003] [SWS_CRYPT_30010] [SWS_CRYPT_30011] [SWS_CRYPT_30098] [SWS_CRYPT_30099] [SWS_CRYPT_30100] [SWS_CRYPT_30101] [SWS_CRYPT_30110] [SWS_CRYPT_30115] [SWS_CRYPT_30123] [SWS_CRYPT_30124] [SWS_CRYPT_30125] [SWS_CRYPT_30126] [SWS_CRYPT_30130] [SWS_CRYPT_30131] [SWS_CRYPT_30200] [SWS_CRYPT_30201] [SWS_CRYPT_30202] [SWS_CRYPT_30203] [SWS_CRYPT_30204] [SWS_CRYPT_30205] [SWS_CRYPT_30206] [SWS_CRYPT_30207] [SWS_CRYPT_30208] [SWS_CRYPT_30209] [SWS_CRYPT_30210] [SWS_CRYPT_30211] [SWS_CRYPT_30212] [SWS_CRYPT_30213] [SWS_CRYPT_30214] [SWS_CRYPT_30215] [SWS_CRYPT_30216] [SWS_CRYPT_30217] [SWS_CRYPT_30218] [SWS_CRYPT_30219] [SWS_CRYPT_30220] [SWS_CRYPT_30221] [SWS_CRYPT_30222] [SWS_CRYPT_30223] [SWS_CRYPT_30224] [SWS_CRYPT_30225] [SWS_CRYPT_30226] [SWS_CRYPT_30227] [SWS_CRYPT_30300] [SWS_CRYPT_30301] [SWS_CRYPT_30302] [SWS_CRYPT_30305] [SWS_CRYPT_30306] [SWS_CRYPT_30307] [SWS_CRYPT_30308] [SWS_CRYPT_30309] [SWS_CRYPT_30310] [SWS_CRYPT_30311] [SWS_CRYPT_30312] [SWS_CRYPT_30313] [SWS_CRYPT_30350] [SWS_CRYPT_30351] [SWS_CRYPT_30400] [SWS_CRYPT_30401] [SWS_CRYPT_30402] [SWS_CRYPT_30403] [SWS_CRYPT_30404] [SWS_CRYPT_30405] [SWS_CRYPT_30406] [SWS_CRYPT_30407] [SWS_CRYPT_30408] [SWS_CRYPT_30409] [SWS_CRYPT_30500] [SWS_CRYPT_30501] [SWS_CRYPT_30503] [SWS_CRYPT_30505] [SWS_CRYPT_30506] [SWS_CRYPT_30508] [SWS_CRYPT_30510] [SWS_CRYPT_30511] [SWS_CRYPT_30550] [SWS_CRYPT_30551] [SWS_CRYPT_40001] [SWS_CRYPT_40002] [SWS_CRYPT_40099] [SWS_CRYPT_40100] [SWS_CRYPT_40101] [SWS_CRYPT_40111] [SWS_CRYPT_40112] [SWS_CRYPT_40113] [SWS_CRYPT_40114] [SWS_CRYPT_40115] [SWS_CRYPT_40150] [SWS_CRYPT_40151] [SWS_CRYPT_40152] [SWS_CRYPT_40153] [SWS_CRYPT_40154] [SWS_CRYPT_40155] [SWS_CRYPT_40156] [SWS_CRYPT_40157] [SWS_CRYPT_40158] [SWS_CRYPT_40159] [SWS_CRYPT_40200] [SWS_CRYPT_40201] [SWS_CRYPT_40202] [SWS_CRYPT_40203] [SWS_CRYPT_40211] [SWS_CRYPT_40212] [SWS_CRYPT_40213] [SWS_CRYPT_40214] [SWS_CRYPT_40215] [SWS_CRYPT_40216] [SWS_CRYPT_40217] [SWS_CRYPT_40218] [SWS_CRYPT_40219] [SWS_CRYPT_40220] [SWS_CRYPT_40221] [SWS_CRYPT_40300] [SWS_CRYPT_40301] [SWS_CRYPT_40302] [SWS_CRYPT_40311] [SWS_CRYPT_40313] [SWS_CRYPT_40314] [SWS_CRYPT_40315] [SWS_CRYPT_40400] [SWS_CRYPT_40401] [SWS_CRYPT_40402] [SWS_CRYPT_40403] [SWS_CRYPT_40411] [SWS_CRYPT_40412] [SWS_CRYPT_40413] [SWS_CRYPT_40414] [SWS_CRYPT_40415] [SWS_CRYPT_40416] [SWS_CRYPT_40417] [SWS_CRYPT_40418] [SWS_CRYPT_40500] [SWS_

CRYPT_40501] [SWS_CRYPT_40511] [SWS_CRYPT_40600] [SWS_CRYPT_40601] [SWS_CRYPT_40602] [SWS_CRYPT_40603] [SWS_CRYPT_40604] [SWS_CRYPT_40611] [SWS_CRYPT_40612] [SWS_CRYPT_40613] [SWS_CRYPT_40614] [SWS_CRYPT_40615] [SWS_CRYPT_40616] [SWS_CRYPT_40617] [SWS_CRYPT_40618] [SWS_CRYPT_40619] [SWS_CRYPT_40620] [SWS_CRYPT_40621] [SWS_CRYPT_40622] [SWS_CRYPT_40624] [SWS_CRYPT_40625] [SWS_CRYPT_40626] [SWS_CRYPT_40627] [SWS_CRYPT_40628] [SWS_CRYPT_40629] [SWS_CRYPT_40630] [SWS_CRYPT_40631] [SWS_CRYPT_40632] [SWS_CRYPT_40633] [SWS_CRYPT_40634] [SWS_CRYPT_40635] [SWS_CRYPT_40636] [SWS_CRYPT_40637] [SWS_CRYPT_40638] [SWS_CRYPT_40639] [SWS_CRYPT_40640] [SWS_CRYPT_40641] [SWS_CRYPT_40700] [SWS_CRYPT_40701] [SWS_CRYPT_40702] [SWS_CRYPT_40711] [SWS_CRYPT_40800] [SWS_CRYPT_40801] [SWS_CRYPT_40802] [SWS_CRYPT_40811] [SWS_CRYPT_40900] [SWS_CRYPT_40911] [SWS_CRYPT_40912] [SWS_CRYPT_40913] [SWS_CRYPT_40914] [SWS_CRYPT_40915] [SWS_CRYPT_40916] [SWS_CRYPT_40917] [SWS_CRYPT_40918] [SWS_CRYPT_40919] [SWS_CRYPT_40920] [SWS_CRYPT_40921] [SWS_CRYPT_40922] [SWS_CRYPT_40923] [SWS_CRYPT_40924] [SWS_CRYPT_40925] [SWS_CRYPT_40926] [SWS_CRYPT_40927] [SWS_CRYPT_40928] [SWS_CRYPT_40929] [SWS_CRYPT_40930] [SWS_CRYPT_40931] [SWS_CRYPT_40932] [SWS_CRYPT_40933] [SWS_CRYPT_40934] [SWS_CRYPT_40935] [SWS_CRYPT_40936] [SWS_CRYPT_40937] [SWS_CRYPT_40938] [SWS_CRYPT_40939] [SWS_CRYPT_40940] [SWS_CRYPT_40941] [SWS_CRYPT_40942] [SWS_CRYPT_40943] [SWS_CRYPT_40944] [SWS_CRYPT_40945] [SWS_CRYPT_40946] [SWS_CRYPT_40947] [SWS_CRYPT_40948] [SWS_CRYPT_40949] [SWS_CRYPT_40950] [SWS_CRYPT_40951] [SWS_CRYPT_40952] [SWS_CRYPT_40953] [SWS_CRYPT_40954] [SWS_CRYPT_40955] [SWS_CRYPT_40956] [SWS_CRYPT_40957] [SWS_CRYPT_40958] [SWS_CRYPT_40959] [SWS_CRYPT_40960] [SWS_CRYPT_40961] [SWS_CRYPT_40962] [SWS_CRYPT_40963] [SWS_CRYPT_40964] [SWS_CRYPT_40965] [SWS_CRYPT_40966] [SWS_CRYPT_40967] [SWS_CRYPT_40968] [SWS_CRYPT_40969] [SWS_CRYPT_40970] [SWS_CRYPT_40971] [SWS_CRYPT_40972] [SWS_CRYPT_40973] [SWS_CRYPT_40974] [SWS_CRYPT_40975] [SWS_CRYPT_40976] [SWS_CRYPT_40977] [SWS_CRYPT_40978] [SWS_CRYPT_40979] [SWS_CRYPT_40980] [SWS_CRYPT_40981] [SWS_CRYPT_40983] [SWS_CRYPT_40984] [SWS_CRYPT_40985]

C.1.2 Changed Specification Items in R22-11

none

C.1.3 Deleted Specification Items in R22-11

none

C.2 Change History of this document according to AUTOSAR Release R23-11

C.2.1 Added Specification Items in R23-11

[SWS_CRYPT_19952] [SWS_CRYPT_40986] [SWS_CRYPT_40987] [SWS_CRYPT_40988] [SWS_CRYPT_40989] [SWS_CRYPT_40990] [SWS_CRYPT_40991] [SWS_CRYPT_40992] [SWS_CRYPT_40993] [SWS_CRYPT_40994] [SWS_CRYPT_40995] [SWS_CRYPT_40996] [SWS_CRYPT_40997] [SWS_CRYPT_40998] [SWS_CRYPT_40999] [SWS_CRYPT_41000] [SWS_CRYPT_41001] [SWS_CRYPT_41002] [SWS_CRYPT_41003] [SWS_CRYPT_41004] [SWS_CRYPT_41005] [SWS_CRYPT_41006] [SWS_CRYPT_41007] [SWS_CRYPT_41008] [SWS_CRYPT_41009] [SWS_CRYPT_41010] [SWS_CRYPT_41011] [SWS_CRYPT_41012] [SWS_CRYPT_41013] [SWS_CRYPT_41014] [SWS_CRYPT_41015] [SWS_CRYPT_41016] [SWS_CRYPT_41017] [SWS_CRYPT_41018]

C.2.2 Changed Specification Items in R23-11

[SWS_CRYPT_01207] [SWS_CRYPT_01211] [SWS_CRYPT_01502] [SWS_CRYPT_01503] [SWS_CRYPT_01659] [SWS_CRYPT_01807] [SWS_CRYPT_02122] [SWS_CRYPT_04204] [SWS_CRYPT_10000] [SWS_CRYPT_10003] [SWS_CRYPT_10005] [SWS_CRYPT_10099] [SWS_CRYPT_10114] [SWS_CRYPT_10305] [SWS_CRYPT_10306] [SWS_CRYPT_10811] [SWS_CRYPT_20721] [SWS_CRYPT_20722] [SWS_CRYPT_20723] [SWS_CRYPT_20730] [SWS_CRYPT_21115] [SWS_CRYPT_21116] [SWS_CRYPT_21519] [SWS_CRYPT_21523] [SWS_CRYPT_22115] [SWS_CRYPT_22116] [SWS_CRYPT_22118] [SWS_CRYPT_22215] [SWS_CRYPT_22911] [SWS_CRYPT_22912] [SWS_CRYPT_22913] [SWS_CRYPT_22914] [SWS_CRYPT_23215] [SWS_CRYPT_23618] [SWS_CRYPT_23623] [SWS_CRYPT_23710] [SWS_CRYPT_23911] [SWS_CRYPT_24018] [SWS_CRYPT_30099] [SWS_CRYPT_30202] [SWS_CRYPT_30203] [SWS_CRYPT_30204] [SWS_CRYPT_30205] [SWS_CRYPT_30206] [SWS_CRYPT_30207] [SWS_CRYPT_30208] [SWS_CRYPT_30209] [SWS_CRYPT_30212] [SWS_CRYPT_30213] [SWS_CRYPT_30214] [SWS_CRYPT_30215] [SWS_CRYPT_30216] [SWS_CRYPT_30217] [SWS_CRYPT_30218] [SWS_CRYPT_30219] [SWS_CRYPT_30220] [SWS_CRYPT_30221] [SWS_CRYPT_30222] [SWS_CRYPT_30223] [SWS_CRYPT_30224] [SWS_CRYPT_30225] [SWS_CRYPT_30226] [SWS_CRYPT_30227] [SWS_CRYPT_40099] [SWS_CRYPT_40203] [SWS_CRYPT_40216] [SWS_CRYPT_40217] [SWS_CRYPT_40218] [SWS_CRYPT_40600] [SWS_CRYPT_40614] [SWS_CRYPT_40616] [SWS_CRYPT_40617] [SWS_CRYPT_40618] [SWS_CRYPT_40619] [SWS_CRYPT_40620] [SWS_CRYPT_40621] [SWS_CRYPT_40622] [SWS_CRYPT_40628] [SWS_CRYPT_40629] [SWS_CRYPT_40630] [SWS_CRYPT_40944] [SWS_CRYPT_40945] [SWS_CRYPT_40962]

C.2.3 Deleted Specification Items in R23-11

[SWS_CRYPT_00622] [SWS_CRYPT_10042] [SWS_CRYPT_10300] [SWS_CRYPT_10301] [SWS_CRYPT_10303] [SWS_CRYPT_10304] [SWS_CRYPT_10712] [SWS_CRYPT_20813] [SWS_CRYPT_21013] [SWS_CRYPT_21117] [SWS_CRYPT_22117] [SWS_CRYPT_22216] [SWS_CRYPT_22713] [SWS_CRYPT_23216] [SWS_CRYPT_23514] [SWS_CRYPT_23617] [SWS_CRYPT_23619] [SWS_CRYPT_23717] [SWS_CRYPT_24017] [SWS_CRYPT_40212] [SWS_CRYPT_40219] [SWS_CRYPT_40221] [SWS_CRYPT_40602] [SWS_CRYPT_40603] [SWS_CRYPT_40624] [SWS_CRYPT_40625] [SWS_CRYPT_40637] [SWS_CRYPT_40638] [SWS_CRYPT_40639]

C.3 Change History of this document according to AUTOSAR Release R24-11

C.3.1 Added Specification Items in R24-11

[SWS_CRYPT_41019] [SWS_CRYPT_41020] [SWS_CRYPT_41021] [SWS_CRYPT_41022] [SWS_CRYPT_41023] [SWS_CRYPT_41024] [SWS_CRYPT_41025] [SWS_CRYPT_41026] [SWS_CRYPT_41027] [SWS_CRYPT_41028] [SWS_CRYPT_41029] [SWS_CRYPT_41030] [SWS_CRYPT_41031] [SWS_CRYPT_41032] [SWS_CRYPT_41033] [SWS_CRYPT_41034] [SWS_CRYPT_41035] [SWS_CRYPT_41036] [SWS_CRYPT_41037] [SWS_CRYPT_41038] [SWS_CRYPT_41039] [SWS_CRYPT_41040] [SWS_CRYPT_41041] [SWS_CRYPT_41042] [SWS_CRYPT_41043] [SWS_CRYPT_41044] [SWS_CRYPT_41045] [SWS_CRYPT_41046] [SWS_CRYPT_41047] [SWS_CRYPT_41048] [SWS_CRYPT_41049] [SWS_CRYPT_41050] [SWS_CRYPT_41051] [SWS_CRYPT_41052] [SWS_CRYPT_41053] [SWS_CRYPT_41054] [SWS_CRYPT_41055] [SWS_CRYPT_41056] [SWS_CRYPT_41057] [SWS_CRYPT_41058]

C.3.2 Changed Specification Items in R24-11

[SWS_CRYPT_00500] [SWS_CRYPT_00501] [SWS_CRYPT_00504] [SWS_CRYPT_00505] [SWS_CRYPT_00506] [SWS_CRYPT_00611] [SWS_CRYPT_00906] [SWS_CRYPT_00907] [SWS_CRYPT_00919] [SWS_CRYPT_01201] [SWS_CRYPT_01210] [SWS_CRYPT_01503] [SWS_CRYPT_01504] [SWS_CRYPT_01653] [SWS_CRYPT_01654] [SWS_CRYPT_01655] [SWS_CRYPT_01656] [SWS_CRYPT_01657] [SWS_CRYPT_01800] [SWS_CRYPT_01804] [SWS_CRYPT_01805] [SWS_CRYPT_01807] [SWS_CRYPT_01808] [SWS_CRYPT_01811] [SWS_CRYPT_02105] [SWS_CRYPT_02109] [SWS_CRYPT_02415] [SWS_CRYPT_02416] [SWS_CRYPT_02417] [SWS_CRYPT_02419] [SWS_CRYPT_02420] [SWS_CRYPT_02422] [SWS_CRYPT_02704] [SWS_CRYPT_02705] [SWS_CRYPT_02726] [SWS_CRYPT_03002] [SWS_CRYPT_03003] [SWS_CRYPT_04202] [SWS_CRYPT_04203] [SWS_CRYPT_04204] [SWS_CRYPT_04213] [SWS_CRYPT_

10003] [SWS_CRYPT_10016] [SWS_CRYPT_10099] [SWS_CRYPT_10111] [SWS_CRYPT_10112] [SWS_CRYPT_10113] [SWS_CRYPT_10114] [SWS_CRYPT_10115] [SWS_CRYPT_10150] [SWS_CRYPT_10151] [SWS_CRYPT_10152] [SWS_CRYPT_10153] [SWS_CRYPT_10154] [SWS_CRYPT_10155] [SWS_CRYPT_10411] [SWS_CRYPT_10451] [SWS_CRYPT_10452] [SWS_CRYPT_10453] [SWS_CRYPT_10454] [SWS_CRYPT_10455] [SWS_CRYPT_10456] [SWS_CRYPT_10701] [SWS_CRYPT_10710] [SWS_CRYPT_10711] [SWS_CRYPT_10808] [SWS_CRYPT_10810] [SWS_CRYPT_10811] [SWS_CRYPT_10812] [SWS_CRYPT_10813] [SWS_CRYPT_10814] [SWS_CRYPT_10815] [SWS_CRYPT_10816] [SWS_CRYPT_10817] [SWS_CRYPT_10818] [SWS_CRYPT_10819] [SWS_CRYPT_10821] [SWS_CRYPT_10822] [SWS_CRYPT_10823] [SWS_CRYPT_10851] [SWS_CRYPT_10853] [SWS_CRYPT_19902] [SWS_CRYPT_19906] [SWS_CRYPT_19950] [SWS_CRYPT_19951] [SWS_CRYPT_19952] [SWS_CRYPT_19953] [SWS_CRYPT_19954] [SWS_CRYPT_20098] [SWS_CRYPT_20099] [SWS_CRYPT_20102] [SWS_CRYPT_20103] [SWS_CRYPT_20312] [SWS_CRYPT_20313] [SWS_CRYPT_20314] [SWS_CRYPT_20316] [SWS_CRYPT_20401] [SWS_CRYPT_20411] [SWS_CRYPT_20412] [SWS_CRYPT_20414] [SWS_CRYPT_20503] [SWS_CRYPT_20505] [SWS_CRYPT_20512] [SWS_CRYPT_20513] [SWS_CRYPT_20514] [SWS_CRYPT_20515] [SWS_CRYPT_20516] [SWS_CRYPT_20517] [SWS_CRYPT_20518] [SWS_CRYPT_20651] [SWS_CRYPT_20652] [SWS_CRYPT_20654] [SWS_CRYPT_20710] [SWS_CRYPT_20711] [SWS_CRYPT_20712] [SWS_CRYPT_20721] [SWS_CRYPT_20722] [SWS_CRYPT_20723] [SWS_CRYPT_20724] [SWS_CRYPT_20725] [SWS_CRYPT_20726] [SWS_CRYPT_20727] [SWS_CRYPT_20728] [SWS_CRYPT_20729] [SWS_CRYPT_20730] [SWS_CRYPT_20731] [SWS_CRYPT_20732] [SWS_CRYPT_20733] [SWS_CRYPT_20741] [SWS_CRYPT_20742] [SWS_CRYPT_20743] [SWS_CRYPT_20744] [SWS_CRYPT_20745] [SWS_CRYPT_20746] [SWS_CRYPT_20747] [SWS_CRYPT_20748] [SWS_CRYPT_20750] [SWS_CRYPT_20751] [SWS_CRYPT_20752] [SWS_CRYPT_20753] [SWS_CRYPT_20754] [SWS_CRYPT_20755] [SWS_CRYPT_20756] [SWS_CRYPT_20757] [SWS_CRYPT_20758] [SWS_CRYPT_20762] [SWS_CRYPT_20763] [SWS_CRYPT_20764] [SWS_CRYPT_20765] [SWS_CRYPT_20802] [SWS_CRYPT_20810] [SWS_CRYPT_20811] [SWS_CRYPT_20812] [SWS_CRYPT_21002] [SWS_CRYPT_21010] [SWS_CRYPT_21011] [SWS_CRYPT_21012] [SWS_CRYPT_21102] [SWS_CRYPT_21110] [SWS_CRYPT_21111] [SWS_CRYPT_21112] [SWS_CRYPT_21113] [SWS_CRYPT_21114] [SWS_CRYPT_21115] [SWS_CRYPT_21116] [SWS_CRYPT_21118] [SWS_CRYPT_21302] [SWS_CRYPT_21311] [SWS_CRYPT_21312] [SWS_CRYPT_21313] [SWS_CRYPT_21314] [SWS_CRYPT_21315] [SWS_CRYPT_21402] [SWS_CRYPT_21411] [SWS_CRYPT_21412] [SWS_CRYPT_21413] [SWS_CRYPT_21414] [SWS_CRYPT_21415] [SWS_CRYPT_21416] [SWS_CRYPT_21512] [SWS_CRYPT_21513] [SWS_CRYPT_21514] [SWS_CRYPT_21515] [SWS_CRYPT_21516] [SWS_CRYPT_21517] [SWS_CRYPT_21519] [SWS_CRYPT_21520] [SWS_CRYPT_21521] [SWS_CRYPT_21522] [SWS_CRYPT_21523] [SWS_CRYPT_21524] [SWS_CRYPT_21525] [SWS_CRYPT_21715] [SWS_CRYPT_21802] [SWS_CRYPT_21810] [SWS_CRYPT_21813] [SWS_CRYPT_21815] [SWS_CRYPT_21816] [SWS_CRYPT_21817] [SWS_CRYPT_21818] [SWS_CRYPT_22102] [SWS_CRYPT_22110] [SWS_CRYPT_

22111] [SWS_CRYPT_22112] [SWS_CRYPT_22113] [SWS_CRYPT_22114] [SWS_CRYPT_22115] [SWS_CRYPT_22116] [SWS_CRYPT_22118] [SWS_CRYPT_22120] [SWS_CRYPT_22210] [SWS_CRYPT_22211] [SWS_CRYPT_22212] [SWS_CRYPT_22213] [SWS_CRYPT_22214] [SWS_CRYPT_22215] [SWS_CRYPT_22511] [SWS_CRYPT_22711] [SWS_CRYPT_22712] [SWS_CRYPT_22902] [SWS_CRYPT_22911] [SWS_CRYPT_22912] [SWS_CRYPT_22913] [SWS_CRYPT_22914] [SWS_CRYPT_22915] [SWS_CRYPT_23011] [SWS_CRYPT_23012] [SWS_CRYPT_23013] [SWS_CRYPT_23014] [SWS_CRYPT_23015] [SWS_CRYPT_23016] [SWS_CRYPT_23210] [SWS_CRYPT_23211] [SWS_CRYPT_23212] [SWS_CRYPT_23213] [SWS_CRYPT_23214] [SWS_CRYPT_23215] [SWS_CRYPT_23510] [SWS_CRYPT_23511] [SWS_CRYPT_23512] [SWS_CRYPT_23513] [SWS_CRYPT_23515] [SWS_CRYPT_23516] [SWS_CRYPT_23602] [SWS_CRYPT_23611] [SWS_CRYPT_23612] [SWS_CRYPT_23613] [SWS_CRYPT_23614] [SWS_CRYPT_23615] [SWS_CRYPT_23616] [SWS_CRYPT_23618] [SWS_CRYPT_23620] [SWS_CRYPT_23621] [SWS_CRYPT_23622] [SWS_CRYPT_23623] [SWS_CRYPT_23624] [SWS_CRYPT_23625] [SWS_CRYPT_23626] [SWS_CRYPT_23627] [SWS_CRYPT_23634] [SWS_CRYPT_23635] [SWS_CRYPT_23702] [SWS_CRYPT_23710] [SWS_CRYPT_23711] [SWS_CRYPT_23712] [SWS_CRYPT_23715] [SWS_CRYPT_23716] [SWS_CRYPT_23911] [SWS_CRYPT_24002] [SWS_CRYPT_24011] [SWS_CRYPT_24012] [SWS_CRYPT_24013] [SWS_CRYPT_24014] [SWS_CRYPT_24015] [SWS_CRYPT_24016] [SWS_CRYPT_24018] [SWS_CRYPT_24019] [SWS_CRYPT_24102] [SWS_CRYPT_24111] [SWS_CRYPT_24112] [SWS_CRYPT_24114] [SWS_CRYPT_24115] [SWS_CRYPT_24116] [SWS_CRYPT_24410] [SWS_CRYPT_24411] [SWS_CRYPT_24412] [SWS_CRYPT_24413] [SWS_CRYPT_24414] [SWS_CRYPT_24415] [SWS_CRYPT_24714] [SWS_CRYPT_24715] [SWS_CRYPT_24811] [SWS_CRYPT_29002] [SWS_CRYPT_29003] [SWS_CRYPT_29004] [SWS_CRYPT_29012] [SWS_CRYPT_29013] [SWS_CRYPT_29014] [SWS_CRYPT_29015] [SWS_CRYPT_29021] [SWS_CRYPT_29022] [SWS_CRYPT_29023] [SWS_CRYPT_29032] [SWS_CRYPT_29033] [SWS_CRYPT_29034] [SWS_CRYPT_29035] [SWS_CRYPT_29041] [SWS_CRYPT_29043] [SWS_CRYPT_29044] [SWS_CRYPT_29045] [SWS_CRYPT_29046] [SWS_CRYPT_29047] [SWS_CRYPT_29048] [SWS_CRYPT_29049] [SWS_CRYPT_30098] [SWS_CRYPT_30099] [SWS_CRYPT_30110] [SWS_CRYPT_30115] [SWS_CRYPT_30123] [SWS_CRYPT_30124] [SWS_CRYPT_30125] [SWS_CRYPT_30130] [SWS_CRYPT_30202] [SWS_CRYPT_30203] [SWS_CRYPT_30204] [SWS_CRYPT_30205] [SWS_CRYPT_30206] [SWS_CRYPT_30207] [SWS_CRYPT_30208] [SWS_CRYPT_30209] [SWS_CRYPT_30210] [SWS_CRYPT_30211] [SWS_CRYPT_30212] [SWS_CRYPT_30213] [SWS_CRYPT_30214] [SWS_CRYPT_30215] [SWS_CRYPT_30216] [SWS_CRYPT_30217] [SWS_CRYPT_30218] [SWS_CRYPT_30219] [SWS_CRYPT_30220] [SWS_CRYPT_30221] [SWS_CRYPT_30222] [SWS_CRYPT_30223] [SWS_CRYPT_30224] [SWS_CRYPT_30225] [SWS_CRYPT_30226] [SWS_CRYPT_30227] [SWS_CRYPT_30301] [SWS_CRYPT_30306] [SWS_CRYPT_30350] [SWS_CRYPT_30351] [SWS_CRYPT_30401] [SWS_CRYPT_30403] [SWS_CRYPT_30404] [SWS_CRYPT_30405] [SWS_CRYPT_30406] [SWS_CRYPT_30407] [SWS_CRYPT_30408] [SWS_CRYPT_30409] [SWS_

CRYPT_30510] [SWS_CRYPT_30550] [SWS_CRYPT_30551] [SWS_CRYPT_40099] [SWS_CRYPT_40111] [SWS_CRYPT_40112] [SWS_CRYPT_40113] [SWS_CRYPT_40114] [SWS_CRYPT_40115] [SWS_CRYPT_40211] [SWS_CRYPT_40214] [SWS_CRYPT_40215] [SWS_CRYPT_40216] [SWS_CRYPT_40217] [SWS_CRYPT_40218] [SWS_CRYPT_40220] [SWS_CRYPT_40311] [SWS_CRYPT_40313] [SWS_CRYPT_40314] [SWS_CRYPT_40411] [SWS_CRYPT_40412] [SWS_CRYPT_40413] [SWS_CRYPT_40414] [SWS_CRYPT_40415] [SWS_CRYPT_40416] [SWS_CRYPT_40417] [SWS_CRYPT_40418] [SWS_CRYPT_40511] [SWS_CRYPT_40604] [SWS_CRYPT_40611] [SWS_CRYPT_40612] [SWS_CRYPT_40613] [SWS_CRYPT_40614] [SWS_CRYPT_40615] [SWS_CRYPT_40616] [SWS_CRYPT_40617] [SWS_CRYPT_40618] [SWS_CRYPT_40619] [SWS_CRYPT_40620] [SWS_CRYPT_40621] [SWS_CRYPT_40622] [SWS_CRYPT_40626] [SWS_CRYPT_40627] [SWS_CRYPT_40628] [SWS_CRYPT_40629] [SWS_CRYPT_40630] [SWS_CRYPT_40631] [SWS_CRYPT_40632] [SWS_CRYPT_40633] [SWS_CRYPT_40634] [SWS_CRYPT_40635] [SWS_CRYPT_40636] [SWS_CRYPT_40640] [SWS_CRYPT_40641] [SWS_CRYPT_40711] [SWS_CRYPT_40811] [SWS_CRYPT_40911] [SWS_CRYPT_40914] [SWS_CRYPT_40915] [SWS_CRYPT_40916] [SWS_CRYPT_40917] [SWS_CRYPT_40918] [SWS_CRYPT_40919] [SWS_CRYPT_40920] [SWS_CRYPT_40921] [SWS_CRYPT_40922] [SWS_CRYPT_40923] [SWS_CRYPT_40924] [SWS_CRYPT_40925] [SWS_CRYPT_40926] [SWS_CRYPT_40927] [SWS_CRYPT_40928] [SWS_CRYPT_40929] [SWS_CRYPT_40930] [SWS_CRYPT_40931] [SWS_CRYPT_40951] [SWS_CRYPT_40970] [SWS_CRYPT_40981] [SWS_CRYPT_40983] [SWS_CRYPT_40995] [SWS_CRYPT_40996] [SWS_CRYPT_40997] [SWS_CRYPT_40998] [SWS_CRYPT_40999] [SWS_CRYPT_41000] [SWS_CRYPT_41001] [SWS_CRYPT_41002] [SWS_CRYPT_41003] [SWS_CRYPT_41004] [SWS_CRYPT_41005] [SWS_CRYPT_41006] [SWS_CRYPT_41007] [SWS_CRYPT_41008] [SWS_CRYPT_41009] [SWS_CRYPT_41010] [SWS_CRYPT_41011] [SWS_CRYPT_41012] [SWS_CRYPT_41013] [SWS_CRYPT_41014] [SWS_CRYPT_41015] [SWS_CRYPT_41016] [SWS_CRYPT_41017] [SWS_CRYPT_41018]

C.3.3 Deleted Specification Items in R24-11

[SWS_CRYPT_00104] [SWS_CRYPT_01208] [SWS_CRYPT_01209] [SWS_CRYPT_01213] [SWS_CRYPT_02421] [SWS_CRYPT_02706] [SWS_CRYPT_10005] [SWS_CRYPT_10750] [SWS_CRYPT_10751] [SWS_CRYPT_10752] [SWS_CRYPT_10753] [SWS_CRYPT_13000] [SWS_CRYPT_13001] [SWS_CRYPT_13002] [SWS_CRYPT_13003] [SWS_CRYPT_20319] [SWS_CRYPT_20760] [SWS_CRYPT_20761] [SWS_CRYPT_22119] [SWS_CRYPT_23300] [SWS_CRYPT_23301] [SWS_CRYPT_23302] [SWS_CRYPT_23311] [SWS_CRYPT_23312] [SWS_CRYPT_24113] [SWS_CRYPT_30126] [SWS_CRYPT_30131] [SWS_CRYPT_40315] [SWS_CRYPT_40960] [SWS_CRYPT_40961] [SWS_CRYPT_40986] [SWS_CRYPT_40987]

C.3.4 Added Constraints in R24-11

[SWS_CRYPT_CONSTR_00001]

C.3.5 Changed Constraints in R24-11

none

C.3.6 Deleted Constraints in R24-11

none

C.4 Change History of this document according to AUTOSAR Release R25-11

C.4.1 Added Specification Items in R25-11

[SWS_CRYPT_04210] [SWS_CRYPT_04211] [SWS_CRYPT_10001] [SWS_CRYPT_13150] [SWS_CRYPT_20620] [SWS_CRYPT_20955] [SWS_CRYPT_41059] [SWS_CRYPT_41060] [SWS_CRYPT_41061] [SWS_CRYPT_41062] [SWS_CRYPT_41063] [SWS_CRYPT_41064] [SWS_CRYPT_41065] [SWS_CRYPT_41066] [SWS_CRYPT_50000] [SWS_CRYPT_50001] [SWS_CRYPT_50002] [SWS_CRYPT_50003] [SWS_CRYPT_50004] [SWS_CRYPT_50005] [SWS_CRYPT_50006] [SWS_CRYPT_50007] [SWS_CRYPT_50008] [SWS_CRYPT_50009] [SWS_CRYPT_50010] [SWS_CRYPT_50011] [SWS_CRYPT_50012] [SWS_CRYPT_50013] [SWS_CRYPT_50014] [SWS_CRYPT_50015] [SWS_CRYPT_50016] [SWS_CRYPT_50017] [SWS_CRYPT_50018] [SWS_CRYPT_50019] [SWS_CRYPT_50020] [SWS_CRYPT_50021] [SWS_CRYPT_50022] [SWS_CRYPT_50023] [SWS_CRYPT_50024] [SWS_CRYPT_50025] [SWS_CRYPT_50026] [SWS_CRYPT_50027] [SWS_CRYPT_50028] [SWS_CRYPT_50029] [SWS_CRYPT_50030] [SWS_CRYPT_50031] [SWS_CRYPT_50032] [SWS_CRYPT_50033] [SWS_CRYPT_50034] [SWS_CRYPT_50035] [SWS_CRYPT_50036] [SWS_CRYPT_50037] [SWS_CRYPT_50038] [SWS_CRYPT_50039] [SWS_CRYPT_50040] [SWS_CRYPT_50041] [SWS_CRYPT_50042] [SWS_CRYPT_50043] [SWS_CRYPT_50044] [SWS_CRYPT_50045] [SWS_CRYPT_50046] [SWS_CRYPT_50047] [SWS_CRYPT_50048] [SWS_CRYPT_50049] [SWS_CRYPT_50050] [SWS_CRYPT_50051] [SWS_CRYPT_50052] [SWS_CRYPT_50053] [SWS_CRYPT_50054] [SWS_CRYPT_50055] [SWS_CRYPT_50056] [SWS_CRYPT_50057] [SWS_CRYPT_50058] [SWS_CRYPT_50059] [SWS_CRYPT_50060] [SWS_CRYPT_50061] [SWS_CRYPT_50062] [SWS_CRYPT_50063] [SWS_CRYPT_50064] [SWS_CRYPT_50065] [SWS_CRYPT_50066] [SWS_CRYPT_50067] [SWS_CRYPT_50068] [SWS_CRYPT_50069] [SWS_CRYPT_50070] [SWS_CRYPT_50071] [SWS_CRYPT_50072] [SWS_CRYPT_50073] [SWS_CRYPT_50074] [SWS_CRYPT_50075] [SWS_CRYPT_

50076] [SWS_CRYPT_50077] [SWS_CRYPT_50078] [SWS_CRYPT_50079] [SWS_CRYPT_50080] [SWS_CRYPT_50081] [SWS_CRYPT_50082] [SWS_CRYPT_50083] [SWS_CRYPT_50084] [SWS_CRYPT_50085] [SWS_CRYPT_50086] [SWS_CRYPT_50087] [SWS_CRYPT_50088] [SWS_CRYPT_50089] [SWS_CRYPT_50090] [SWS_CRYPT_50091] [SWS_CRYPT_50092] [SWS_CRYPT_50093] [SWS_CRYPT_50094] [SWS_CRYPT_50095] [SWS_CRYPT_50096] [SWS_CRYPT_50097] [SWS_CRYPT_50098] [SWS_CRYPT_50099] [SWS_CRYPT_50100] [SWS_CRYPT_50101] [SWS_CRYPT_50102] [SWS_CRYPT_50103] [SWS_CRYPT_50104] [SWS_CRYPT_50105] [SWS_CRYPT_50106] [SWS_CRYPT_50107] [SWS_CRYPT_50108] [SWS_CRYPT_50109] [SWS_CRYPT_50110] [SWS_CRYPT_50111] [SWS_CRYPT_50112] [SWS_CRYPT_50113] [SWS_CRYPT_50114] [SWS_CRYPT_50115] [SWS_CRYPT_50116] [SWS_CRYPT_50117] [SWS_CRYPT_50118] [SWS_CRYPT_50119] [SWS_CRYPT_50120] [SWS_CRYPT_50121] [SWS_CRYPT_50122] [SWS_CRYPT_50123] [SWS_CRYPT_50124] [SWS_CRYPT_50125] [SWS_CRYPT_50126] [SWS_CRYPT_50127] [SWS_CRYPT_50128] [SWS_CRYPT_50129] [SWS_CRYPT_50130] [SWS_CRYPT_50131] [SWS_CRYPT_50132] [SWS_CRYPT_50133] [SWS_CRYPT_50134] [SWS_CRYPT_50135] [SWS_CRYPT_50136] [SWS_CRYPT_50137] [SWS_CRYPT_50138] [SWS_CRYPT_50139] [SWS_CRYPT_50140] [SWS_CRYPT_50141] [SWS_CRYPT_50142] [SWS_CRYPT_50143] [SWS_CRYPT_50144] [SWS_CRYPT_50145] [SWS_CRYPT_50146] [SWS_CRYPT_50147] [SWS_CRYPT_50148] [SWS_CRYPT_50149] [SWS_CRYPT_50150] [SWS_CRYPT_50151] [SWS_CRYPT_50152] [SWS_CRYPT_50153] [SWS_CRYPT_50154] [SWS_CRYPT_50155] [SWS_CRYPT_50156] [SWS_CRYPT_50157] [SWS_CRYPT_50158] [SWS_CRYPT_50159] [SWS_CRYPT_50160] [SWS_CRYPT_50161] [SWS_CRYPT_50162] [SWS_CRYPT_50163] [SWS_CRYPT_50164] [SWS_CRYPT_50165] [SWS_CRYPT_50166] [SWS_CRYPT_50167] [SWS_CRYPT_50168] [SWS_CRYPT_50169] [SWS_CRYPT_50170] [SWS_CRYPT_50171] [SWS_CRYPT_50172] [SWS_CRYPT_50173] [SWS_CRYPT_50174] [SWS_CRYPT_50175] [SWS_CRYPT_50176] [SWS_CRYPT_50177] [SWS_CRYPT_50178] [SWS_CRYPT_50179] [SWS_CRYPT_50180] [SWS_CRYPT_50181] [SWS_CRYPT_50182] [SWS_CRYPT_50183] [SWS_CRYPT_50184] [SWS_CRYPT_50185] [SWS_CRYPT_50186] [SWS_CRYPT_50187] [SWS_CRYPT_50190] [SWS_CRYPT_50191] [SWS_CRYPT_50192] [SWS_CRYPT_50193] [SWS_CRYPT_50194] [SWS_CRYPT_50195] [SWS_CRYPT_50196] [SWS_CRYPT_50197] [SWS_CRYPT_50198] [SWS_CRYPT_50199] [SWS_CRYPT_50200] [SWS_CRYPT_50205] [SWS_CRYPT_50206] [SWS_CRYPT_50207] [SWS_CRYPT_50208] [SWS_CRYPT_50209] [SWS_CRYPT_50210] [SWS_CRYPT_50211] [SWS_CRYPT_50212] [SWS_CRYPT_50213] [SWS_CRYPT_50214] [SWS_CRYPT_60000] [SWS_CRYPT_60001] [SWS_CRYPT_60002] [SWS_CRYPT_60003] [SWS_CRYPT_60004] [SWS_CRYPT_60005] [SWS_CRYPT_60006] [SWS_CRYPT_60010] [SWS_CRYPT_60011] [SWS_CRYPT_60012] [SWS_CRYPT_60013] [SWS_CRYPT_60014] [SWS_CRYPT_60015] [SWS_CRYPT_60016] [SWS_CRYPT_60017] [SWS_CRYPT_60018] [SWS_CRYPT_60019] [SWS_CRYPT_60020] [SWS_CRYPT_60021] [SWS_CRYPT_60022]

C.4.2 Changed Specification Items in R25-11

[SWS_CRYPT_00101] [SWS_CRYPT_00502] [SWS_CRYPT_03910] [SWS_CRYPT_04203] [SWS_CRYPT_04205] [SWS_CRYPT_10000] [SWS_CRYPT_10014] [SWS_CRYPT_10015] [SWS_CRYPT_10016] [SWS_CRYPT_10019] [SWS_CRYPT_10031] [SWS_CRYPT_10099] [SWS_CRYPT_10701] [SWS_CRYPT_10710] [SWS_CRYPT_10711] [SWS_CRYPT_13101] [SWS_CRYPT_13102] [SWS_CRYPT_13103] [SWS_CRYPT_13104] [SWS_CRYPT_13105] [SWS_CRYPT_13106] [SWS_CRYPT_13107] [SWS_CRYPT_13108] [SWS_CRYPT_13109] [SWS_CRYPT_13110] [SWS_CRYPT_19902] [SWS_CRYPT_19906] [SWS_CRYPT_19950] [SWS_CRYPT_19951] [SWS_CRYPT_19952] [SWS_CRYPT_19953] [SWS_CRYPT_19954] [SWS_CRYPT_20098] [SWS_CRYPT_20099] [SWS_CRYPT_20100] [SWS_CRYPT_20101] [SWS_CRYPT_20103] [SWS_CRYPT_20313] [SWS_CRYPT_20401] [SWS_CRYPT_20411] [SWS_CRYPT_20500] [SWS_CRYPT_20501] [SWS_CRYPT_20505] [SWS_CRYPT_20517] [SWS_CRYPT_20618] [SWS_CRYPT_20700] [SWS_CRYPT_20701] [SWS_CRYPT_20710] [SWS_CRYPT_20721] [SWS_CRYPT_20722] [SWS_CRYPT_20728] [SWS_CRYPT_20732] [SWS_CRYPT_20741] [SWS_CRYPT_20742] [SWS_CRYPT_20743] [SWS_CRYPT_20745] [SWS_CRYPT_20746] [SWS_CRYPT_20747] [SWS_CRYPT_20748] [SWS_CRYPT_20750] [SWS_CRYPT_20751] [SWS_CRYPT_20752] [SWS_CRYPT_20753] [SWS_CRYPT_20756] [SWS_CRYPT_20757] [SWS_CRYPT_20758] [SWS_CRYPT_20762] [SWS_CRYPT_20763] [SWS_CRYPT_20764] [SWS_CRYPT_20907] [SWS_CRYPT_21101] [SWS_CRYPT_21113] [SWS_CRYPT_21115] [SWS_CRYPT_21116] [SWS_CRYPT_21301] [SWS_CRYPT_21312] [SWS_CRYPT_21313] [SWS_CRYPT_21401] [SWS_CRYPT_21412] [SWS_CRYPT_21413] [SWS_CRYPT_21501] [SWS_CRYPT_21515] [SWS_CRYPT_21525] [SWS_CRYPT_21801] [SWS_CRYPT_21813] [SWS_CRYPT_21815] [SWS_CRYPT_22101] [SWS_CRYPT_22110] [SWS_CRYPT_22111] [SWS_CRYPT_22113] [SWS_CRYPT_22115] [SWS_CRYPT_22116] [SWS_CRYPT_22118] [SWS_CRYPT_22500] [SWS_CRYPT_22501] [SWS_CRYPT_22511] [SWS_CRYPT_22700] [SWS_CRYPT_22701] [SWS_CRYPT_22901] [SWS_CRYPT_22911] [SWS_CRYPT_22912] [SWS_CRYPT_22914] [SWS_CRYPT_22915] [SWS_CRYPT_23501] [SWS_CRYPT_23512] [SWS_CRYPT_23515] [SWS_CRYPT_23613] [SWS_CRYPT_23618] [SWS_CRYPT_23625] [SWS_CRYPT_23800] [SWS_CRYPT_23801] [SWS_CRYPT_23911] [SWS_CRYPT_24000] [SWS_CRYPT_24001] [SWS_CRYPT_24014] [SWS_CRYPT_24016] [SWS_CRYPT_24018] [SWS_CRYPT_24101] [SWS_CRYPT_24112] [SWS_CRYPT_24115] [SWS_CRYPT_24715] [SWS_CRYPT_29004] [SWS_CRYPT_30001] [SWS_CRYPT_30002] [SWS_CRYPT_30003] [SWS_CRYPT_30098] [SWS_CRYPT_30403] [SWS_CRYPT_30404] [SWS_CRYPT_30405] [SWS_CRYPT_30406] [SWS_CRYPT_40001] [SWS_CRYPT_40099] [SWS_CRYPT_40101] [SWS_CRYPT_40111] [SWS_CRYPT_40112] [SWS_CRYPT_40115] [SWS_CRYPT_40151] [SWS_CRYPT_40152] [SWS_CRYPT_40153] [SWS_CRYPT_40154] [SWS_CRYPT_40155] [SWS_CRYPT_40156] [SWS_CRYPT_40157] [SWS_CRYPT_40158] [SWS_CRYPT_40159] [SWS_CRYPT_40202] [SWS_CRYPT_40203] [SWS_CRYPT_40213] [SWS_CRYPT_40214] [SWS_CRYPT_40215] [SWS_CRYPT_40216] [SWS_CRYPT_40300] [SWS_CRYPT_40301] [SWS_CRYPT_

40400] [SWS_CRYPT_40401] [SWS_CRYPT_40411] [SWS_CRYPT_40600] [SWS_CRYPT_40601] [SWS_CRYPT_40604] [SWS_CRYPT_40614] [SWS_CRYPT_40616] [SWS_CRYPT_40621] [SWS_CRYPT_40622] [SWS_CRYPT_40626] [SWS_CRYPT_40627] [SWS_CRYPT_40640] [SWS_CRYPT_40641] [SWS_CRYPT_40802] [SWS_CRYPT_40962] [SWS_CRYPT_40990] [SWS_CRYPT_40991] [SWS_CRYPT_40992] [SWS_CRYPT_41019] [SWS_CRYPT_41020] [SWS_CRYPT_41021] [SWS_CRYPT_41022] [SWS_CRYPT_41023] [SWS_CRYPT_41024] [SWS_CRYPT_41043] [SWS_CRYPT_41044] [SWS_CRYPT_41047] [SWS_CRYPT_41050]

C.4.3 Deleted Specification Items in R25-11

[SWS_CRYPT_00005] [SWS_CRYPT_00006] [SWS_CRYPT_00009] [SWS_CRYPT_00500] [SWS_CRYPT_00501] [SWS_CRYPT_00503] [SWS_CRYPT_00504] [SWS_CRYPT_00505] [SWS_CRYPT_00506] [SWS_CRYPT_00507] [SWS_CRYPT_00508] [SWS_CRYPT_00601] [SWS_CRYPT_00603] [SWS_CRYPT_00608] [SWS_CRYPT_00609] [SWS_CRYPT_00610] [SWS_CRYPT_00611] [SWS_CRYPT_00901] [SWS_CRYPT_00902] [SWS_CRYPT_00903] [SWS_CRYPT_00905] [SWS_CRYPT_00906] [SWS_CRYPT_00907] [SWS_CRYPT_00908] [SWS_CRYPT_00909] [SWS_CRYPT_00910] [SWS_CRYPT_00919] [SWS_CRYPT_01200] [SWS_CRYPT_01201] [SWS_CRYPT_01202] [SWS_CRYPT_01203] [SWS_CRYPT_01204] [SWS_CRYPT_01207] [SWS_CRYPT_01210] [SWS_CRYPT_01211] [SWS_CRYPT_01501] [SWS_CRYPT_01502] [SWS_CRYPT_01503] [SWS_CRYPT_01504] [SWS_CRYPT_01506] [SWS_CRYPT_01508] [SWS_CRYPT_01651] [SWS_CRYPT_01653] [SWS_CRYPT_01654] [SWS_CRYPT_01655] [SWS_CRYPT_01656] [SWS_CRYPT_01657] [SWS_CRYPT_01658] [SWS_CRYPT_01659] [SWS_CRYPT_01660] [SWS_CRYPT_01661] [SWS_CRYPT_01662] [SWS_CRYPT_01800] [SWS_CRYPT_01801] [SWS_CRYPT_01802] [SWS_CRYPT_01803] [SWS_CRYPT_01804] [SWS_CRYPT_01805] [SWS_CRYPT_01806] [SWS_CRYPT_01807] [SWS_CRYPT_01808] [SWS_CRYPT_01811] [SWS_CRYPT_01820] [SWS_CRYPT_01821] [SWS_CRYPT_01822] [SWS_CRYPT_01823] [SWS_CRYPT_02104] [SWS_CRYPT_02105] [SWS_CRYPT_02106] [SWS_CRYPT_02107] [SWS_CRYPT_02108] [SWS_CRYPT_02109] [SWS_CRYPT_02121] [SWS_CRYPT_02122] [SWS_CRYPT_02123] [SWS_CRYPT_02400] [SWS_CRYPT_02408] [SWS_CRYPT_02409] [SWS_CRYPT_02410] [SWS_CRYPT_02411] [SWS_CRYPT_02412] [SWS_CRYPT_02413] [SWS_CRYPT_02414] [SWS_CRYPT_02415] [SWS_CRYPT_02416] [SWS_CRYPT_02417] [SWS_CRYPT_02418] [SWS_CRYPT_02419] [SWS_CRYPT_02420] [SWS_CRYPT_02422] [SWS_CRYPT_02700] [SWS_CRYPT_02701] [SWS_CRYPT_02702] [SWS_CRYPT_02703] [SWS_CRYPT_02704] [SWS_CRYPT_02705] [SWS_CRYPT_02726] [SWS_CRYPT_03000] [SWS_CRYPT_03002] [SWS_CRYPT_03003] [SWS_CRYPT_03004] [SWS_CRYPT_03005] [SWS_CRYPT_03006] [SWS_CRYPT_03007] [SWS_CRYPT_03008] [SWS_CRYPT_03009] [SWS_CRYPT_03300] [SWS_CRYPT_03301] [SWS_CRYPT_03302] [SWS_CRYPT_03303] [SWS_CRYPT_03304] [SWS_CRYPT_03305] [SWS_CRYPT_03311] [SWS_CRYPT_03312] [SWS_CRYPT_03313] [SWS_

CRYPT_03904] [SWS_CRYPT_03905] [SWS_CRYPT_03906] [SWS_CRYPT_04200] [SWS_CRYPT_04202] [SWS_CRYPT_04204] [SWS_CRYPT_04207] [SWS_CRYPT_04208] [SWS_CRYPT_04209] [SWS_CRYPT_04213] [SWS_CRYPT_10003] [SWS_CRYPT_10004] [SWS_CRYPT_10017] [SWS_CRYPT_10018] [SWS_CRYPT_10100] [SWS_CRYPT_10101] [SWS_CRYPT_10102] [SWS_CRYPT_10111] [SWS_CRYPT_10112] [SWS_CRYPT_10113] [SWS_CRYPT_10114] [SWS_CRYPT_10115] [SWS_CRYPT_10150] [SWS_CRYPT_10151] [SWS_CRYPT_10152] [SWS_CRYPT_10153] [SWS_CRYPT_10154] [SWS_CRYPT_10155] [SWS_CRYPT_10200] [SWS_CRYPT_10305] [SWS_CRYPT_10306] [SWS_CRYPT_10400] [SWS_CRYPT_10401] [SWS_CRYPT_10402] [SWS_CRYPT_10403] [SWS_CRYPT_10411] [SWS_CRYPT_10412] [SWS_CRYPT_10413] [SWS_CRYPT_10451] [SWS_CRYPT_10452] [SWS_CRYPT_10453] [SWS_CRYPT_10454] [SWS_CRYPT_10455] [SWS_CRYPT_10456] [SWS_CRYPT_10800] [SWS_CRYPT_10801] [SWS_CRYPT_10802] [SWS_CRYPT_10808] [SWS_CRYPT_10810] [SWS_CRYPT_10811] [SWS_CRYPT_10812] [SWS_CRYPT_10813] [SWS_CRYPT_10814] [SWS_CRYPT_10815] [SWS_CRYPT_10816] [SWS_CRYPT_10817] [SWS_CRYPT_10818] [SWS_CRYPT_10819] [SWS_CRYPT_10821] [SWS_CRYPT_10822] [SWS_CRYPT_10823] [SWS_CRYPT_10850] [SWS_CRYPT_10851] [SWS_CRYPT_10852] [SWS_CRYPT_10853] [SWS_CRYPT_13100] [SWS_CRYPT_13111] [SWS_CRYPT_13112] [SWS_CRYPT_13113] [SWS_CRYPT_13114] [SWS_CRYPT_13115] [SWS_CRYPT_13116] [SWS_CRYPT_13117] [SWS_CRYPT_13118] [SWS_CRYPT_13119] [SWS_CRYPT_13120] [SWS_CRYPT_13121] [SWS_CRYPT_13122] [SWS_CRYPT_20000] [SWS_CRYPT_20001] [SWS_CRYPT_20002] [SWS_CRYPT_20003] [SWS_CRYPT_20004] [SWS_CRYPT_20005] [SWS_CRYPT_20006] [SWS_CRYPT_20007] [SWS_CRYPT_20009] [SWS_CRYPT_20010] [SWS_CRYPT_20011] [SWS_CRYPT_20102] [SWS_CRYPT_20301] [SWS_CRYPT_20302] [SWS_CRYPT_20303] [SWS_CRYPT_20304] [SWS_CRYPT_20312] [SWS_CRYPT_20314] [SWS_CRYPT_20316] [SWS_CRYPT_20402] [SWS_CRYPT_20412] [SWS_CRYPT_20414] [SWS_CRYPT_20502] [SWS_CRYPT_20503] [SWS_CRYPT_20504] [SWS_CRYPT_20506] [SWS_CRYPT_20507] [SWS_CRYPT_20512] [SWS_CRYPT_20513] [SWS_CRYPT_20514] [SWS_CRYPT_20515] [SWS_CRYPT_20516] [SWS_CRYPT_20518] [SWS_CRYPT_20600] [SWS_CRYPT_20601] [SWS_CRYPT_20602] [SWS_CRYPT_20603] [SWS_CRYPT_20611] [SWS_CRYPT_20612] [SWS_CRYPT_20613] [SWS_CRYPT_20614] [SWS_CRYPT_20615] [SWS_CRYPT_20616] [SWS_CRYPT_20617] [SWS_CRYPT_20619] [SWS_CRYPT_20641] [SWS_CRYPT_20643] [SWS_CRYPT_20644] [SWS_CRYPT_20651] [SWS_CRYPT_20652] [SWS_CRYPT_20654] [SWS_CRYPT_20703] [SWS_CRYPT_20711] [SWS_CRYPT_20712] [SWS_CRYPT_20723] [SWS_CRYPT_20724] [SWS_CRYPT_20725] [SWS_CRYPT_20726] [SWS_CRYPT_20727] [SWS_CRYPT_20729] [SWS_CRYPT_20730] [SWS_CRYPT_20731] [SWS_CRYPT_20733] [SWS_CRYPT_20744] [SWS_CRYPT_20754] [SWS_CRYPT_20755] [SWS_CRYPT_20765] [SWS_CRYPT_20800] [SWS_CRYPT_20801] [SWS_CRYPT_20802] [SWS_CRYPT_20810] [SWS_CRYPT_20811] [SWS_CRYPT_20812] [SWS_CRYPT_20901] [SWS_CRYPT_20902] [SWS_CRYPT_20903] [SWS_CRYPT_20904] [SWS_CRYPT_20905] [SWS_CRYPT_20906] [SWS_CRYPT_20908] [SWS_CRYPT_20909] [SWS_CRYPT_20910] [SWS_CRYPT_21000] [SWS_

CRYPT_21001] [SWS_CRYPT_21002] [SWS_CRYPT_21010] [SWS_CRYPT_21011] [SWS_CRYPT_21012] [SWS_CRYPT_21102] [SWS_CRYPT_21110] [SWS_CRYPT_21111] [SWS_CRYPT_21112] [SWS_CRYPT_21114] [SWS_CRYPT_21118] [SWS_CRYPT_21302] [SWS_CRYPT_21311] [SWS_CRYPT_21314] [SWS_CRYPT_21315] [SWS_CRYPT_21402] [SWS_CRYPT_21411] [SWS_CRYPT_21414] [SWS_CRYPT_21415] [SWS_CRYPT_21416] [SWS_CRYPT_21512] [SWS_CRYPT_21513] [SWS_CRYPT_21514] [SWS_CRYPT_21516] [SWS_CRYPT_21517] [SWS_CRYPT_21519] [SWS_CRYPT_21520] [SWS_CRYPT_21521] [SWS_CRYPT_21522] [SWS_CRYPT_21523] [SWS_CRYPT_21524] [SWS_CRYPT_21715] [SWS_CRYPT_21802] [SWS_CRYPT_21810] [SWS_CRYPT_21816] [SWS_CRYPT_21817] [SWS_CRYPT_21818] [SWS_CRYPT_22102] [SWS_CRYPT_22112] [SWS_CRYPT_22114] [SWS_CRYPT_22120] [SWS_CRYPT_22200] [SWS_CRYPT_22201] [SWS_CRYPT_22210] [SWS_CRYPT_22211] [SWS_CRYPT_22212] [SWS_CRYPT_22213] [SWS_CRYPT_22214] [SWS_CRYPT_22215] [SWS_CRYPT_22503] [SWS_CRYPT_22702] [SWS_CRYPT_22711] [SWS_CRYPT_22712] [SWS_CRYPT_22902] [SWS_CRYPT_22913] [SWS_CRYPT_23000] [SWS_CRYPT_23001] [SWS_CRYPT_23002] [SWS_CRYPT_23003] [SWS_CRYPT_23011] [SWS_CRYPT_23012] [SWS_CRYPT_23013] [SWS_CRYPT_23014] [SWS_CRYPT_23015] [SWS_CRYPT_23016] [SWS_CRYPT_23200] [SWS_CRYPT_23201] [SWS_CRYPT_23210] [SWS_CRYPT_23211] [SWS_CRYPT_23212] [SWS_CRYPT_23213] [SWS_CRYPT_23214] [SWS_CRYPT_23215] [SWS_CRYPT_23510] [SWS_CRYPT_23511] [SWS_CRYPT_23513] [SWS_CRYPT_23516] [SWS_CRYPT_23600] [SWS_CRYPT_23601] [SWS_CRYPT_23602] [SWS_CRYPT_23611] [SWS_CRYPT_23612] [SWS_CRYPT_23614] [SWS_CRYPT_23615] [SWS_CRYPT_23616] [SWS_CRYPT_23620] [SWS_CRYPT_23621] [SWS_CRYPT_23622] [SWS_CRYPT_23623] [SWS_CRYPT_23624] [SWS_CRYPT_23626] [SWS_CRYPT_23627] [SWS_CRYPT_23634] [SWS_CRYPT_23635] [SWS_CRYPT_23700] [SWS_CRYPT_23701] [SWS_CRYPT_23702] [SWS_CRYPT_23710] [SWS_CRYPT_23711] [SWS_CRYPT_23712] [SWS_CRYPT_23715] [SWS_CRYPT_23716] [SWS_CRYPT_23802] [SWS_CRYPT_24002] [SWS_CRYPT_24011] [SWS_CRYPT_24012] [SWS_CRYPT_24013] [SWS_CRYPT_24015] [SWS_CRYPT_24019] [SWS_CRYPT_24102] [SWS_CRYPT_24111] [SWS_CRYPT_24114] [SWS_CRYPT_24116] [SWS_CRYPT_24400] [SWS_CRYPT_24401] [SWS_CRYPT_24410] [SWS_CRYPT_24411] [SWS_CRYPT_24412] [SWS_CRYPT_24413] [SWS_CRYPT_24414] [SWS_CRYPT_24415] [SWS_CRYPT_24714] [SWS_CRYPT_24800] [SWS_CRYPT_24801] [SWS_CRYPT_24802] [SWS_CRYPT_24811] [SWS_CRYPT_29000] [SWS_CRYPT_29001] [SWS_CRYPT_29002] [SWS_CRYPT_29003] [SWS_CRYPT_29010] [SWS_CRYPT_29011] [SWS_CRYPT_29012] [SWS_CRYPT_29013] [SWS_CRYPT_29014] [SWS_CRYPT_29015] [SWS_CRYPT_29020] [SWS_CRYPT_29021] [SWS_CRYPT_29022] [SWS_CRYPT_29023] [SWS_CRYPT_29024] [SWS_CRYPT_29030] [SWS_CRYPT_29031] [SWS_CRYPT_29032] [SWS_CRYPT_29033] [SWS_CRYPT_29034] [SWS_CRYPT_29035] [SWS_CRYPT_29040] [SWS_CRYPT_29041] [SWS_CRYPT_29042] [SWS_CRYPT_29043] [SWS_CRYPT_29044] [SWS_CRYPT_29045] [SWS_CRYPT_29046] [SWS_CRYPT_29047] [SWS_CRYPT_29048] [SWS_CRYPT_29049] [SWS_CRYPT_30010] [SWS_CRYPT_30011] [SWS_CRYPT_30099] [SWS_CRYPT_30100] [SWS_

CRYPT_30101] [SWS_CRYPT_30110] [SWS_CRYPT_30115] [SWS_CRYPT_30123] [SWS_CRYPT_30124] [SWS_CRYPT_30125] [SWS_CRYPT_30130] [SWS_CRYPT_30200] [SWS_CRYPT_30201] [SWS_CRYPT_30202] [SWS_CRYPT_30203] [SWS_CRYPT_30206] [SWS_CRYPT_30207] [SWS_CRYPT_30208] [SWS_CRYPT_30209] [SWS_CRYPT_30210] [SWS_CRYPT_30211] [SWS_CRYPT_30212] [SWS_CRYPT_30213] [SWS_CRYPT_30218] [SWS_CRYPT_30219] [SWS_CRYPT_30220] [SWS_CRYPT_30221] [SWS_CRYPT_30222] [SWS_CRYPT_30223] [SWS_CRYPT_30224] [SWS_CRYPT_30225] [SWS_CRYPT_30300] [SWS_CRYPT_30301] [SWS_CRYPT_30302] [SWS_CRYPT_30305] [SWS_CRYPT_30306] [SWS_CRYPT_30307] [SWS_CRYPT_30308] [SWS_CRYPT_30309] [SWS_CRYPT_30310] [SWS_CRYPT_30311] [SWS_CRYPT_30312] [SWS_CRYPT_30313] [SWS_CRYPT_30350] [SWS_CRYPT_30351] [SWS_CRYPT_30400] [SWS_CRYPT_30401] [SWS_CRYPT_30402] [SWS_CRYPT_30407] [SWS_CRYPT_30408] [SWS_CRYPT_30409] [SWS_CRYPT_30500] [SWS_CRYPT_30501] [SWS_CRYPT_30503] [SWS_CRYPT_30505] [SWS_CRYPT_30506] [SWS_CRYPT_30508] [SWS_CRYPT_30510] [SWS_CRYPT_30511] [SWS_CRYPT_30550] [SWS_CRYPT_30551] [SWS_CRYPT_40002] [SWS_CRYPT_40100] [SWS_CRYPT_40113] [SWS_CRYPT_40114] [SWS_CRYPT_40150] [SWS_CRYPT_40200] [SWS_CRYPT_40201] [SWS_CRYPT_40211] [SWS_CRYPT_40217] [SWS_CRYPT_40218] [SWS_CRYPT_40220] [SWS_CRYPT_40302] [SWS_CRYPT_40311] [SWS_CRYPT_40313] [SWS_CRYPT_40314] [SWS_CRYPT_40402] [SWS_CRYPT_40403] [SWS_CRYPT_40412] [SWS_CRYPT_40413] [SWS_CRYPT_40414] [SWS_CRYPT_40415] [SWS_CRYPT_40416] [SWS_CRYPT_40417] [SWS_CRYPT_40418] [SWS_CRYPT_40500] [SWS_CRYPT_40501] [SWS_CRYPT_40511] [SWS_CRYPT_40611] [SWS_CRYPT_40612] [SWS_CRYPT_40613] [SWS_CRYPT_40615] [SWS_CRYPT_40617] [SWS_CRYPT_40618] [SWS_CRYPT_40619] [SWS_CRYPT_40620] [SWS_CRYPT_40628] [SWS_CRYPT_40629] [SWS_CRYPT_40630] [SWS_CRYPT_40631] [SWS_CRYPT_40632] [SWS_CRYPT_40633] [SWS_CRYPT_40634] [SWS_CRYPT_40635] [SWS_CRYPT_40636] [SWS_CRYPT_40700] [SWS_CRYPT_40701] [SWS_CRYPT_40702] [SWS_CRYPT_40711] [SWS_CRYPT_40800] [SWS_CRYPT_40801] [SWS_CRYPT_40811] [SWS_CRYPT_40900] [SWS_CRYPT_40911] [SWS_CRYPT_40912] [SWS_CRYPT_40913] [SWS_CRYPT_40914] [SWS_CRYPT_40915] [SWS_CRYPT_40916] [SWS_CRYPT_40917] [SWS_CRYPT_40918] [SWS_CRYPT_40919] [SWS_CRYPT_40920] [SWS_CRYPT_40921] [SWS_CRYPT_40922] [SWS_CRYPT_40923] [SWS_CRYPT_40924] [SWS_CRYPT_40925] [SWS_CRYPT_40926] [SWS_CRYPT_40927] [SWS_CRYPT_40928] [SWS_CRYPT_40929] [SWS_CRYPT_40930] [SWS_CRYPT_40931] [SWS_CRYPT_40932] [SWS_CRYPT_40933] [SWS_CRYPT_40934] [SWS_CRYPT_40935] [SWS_CRYPT_40936] [SWS_CRYPT_40937] [SWS_CRYPT_40938] [SWS_CRYPT_40939] [SWS_CRYPT_40940] [SWS_CRYPT_40941] [SWS_CRYPT_40942] [SWS_CRYPT_40943] [SWS_CRYPT_40944] [SWS_CRYPT_40945] [SWS_CRYPT_40946] [SWS_CRYPT_40947] [SWS_CRYPT_40948] [SWS_CRYPT_40949] [SWS_CRYPT_40950] [SWS_CRYPT_40951] [SWS_CRYPT_40952] [SWS_CRYPT_40953] [SWS_CRYPT_40954] [SWS_CRYPT_40955] [SWS_CRYPT_40956] [SWS_CRYPT_40957] [SWS_CRYPT_40958] [SWS_CRYPT_40959] [SWS_CRYPT_40963] [SWS_CRYPT_40964] [SWS_CRYPT_40965] [SWS_

CRYPT_40966] [SWS_CRYPT_40967] [SWS_CRYPT_40968] [SWS_CRYPT_40969] [SWS_CRYPT_40970] [SWS_CRYPT_40971] [SWS_CRYPT_40972] [SWS_CRYPT_40973] [SWS_CRYPT_40974] [SWS_CRYPT_40975] [SWS_CRYPT_40976] [SWS_CRYPT_40977] [SWS_CRYPT_40978] [SWS_CRYPT_40979] [SWS_CRYPT_40980] [SWS_CRYPT_40981] [SWS_CRYPT_40983] [SWS_CRYPT_40984] [SWS_CRYPT_40985] [SWS_CRYPT_40988] [SWS_CRYPT_40989] [SWS_CRYPT_40993] [SWS_CRYPT_40994] [SWS_CRYPT_40995] [SWS_CRYPT_40996] [SWS_CRYPT_40999] [SWS_CRYPT_41000] [SWS_CRYPT_41001] [SWS_CRYPT_41002] [SWS_CRYPT_41007] [SWS_CRYPT_41008] [SWS_CRYPT_41009] [SWS_CRYPT_41010] [SWS_CRYPT_41011] [SWS_CRYPT_41012] [SWS_CRYPT_41015] [SWS_CRYPT_41016] [SWS_CRYPT_41017] [SWS_CRYPT_41018] [SWS_CRYPT_41025] [SWS_CRYPT_41026] [SWS_CRYPT_41027] [SWS_CRYPT_41028] [SWS_CRYPT_41029] [SWS_CRYPT_41030] [SWS_CRYPT_41031] [SWS_CRYPT_41032] [SWS_CRYPT_41033] [SWS_CRYPT_41034] [SWS_CRYPT_41035] [SWS_CRYPT_41036] [SWS_CRYPT_41037] [SWS_CRYPT_41038] [SWS_CRYPT_41039] [SWS_CRYPT_41040] [SWS_CRYPT_41041] [SWS_CRYPT_41042] [SWS_CRYPT_41046] [SWS_CRYPT_41048] [SWS_CRYPT_41053] [SWS_CRYPT_41054] [SWS_CRYPT_41057]

C.4.4 Added Constraints in R25-11

none

C.4.5 Changed Constraints in R25-11

none

C.4.6 Deleted Constraints in R25-11

none

D Naming Convention

This section proposed names for a set of standardized transformations. A CryptoProvider that implements a transformation that is contained in the following list should use the proposed names. A CryptoProvider implementation may additionally implement algorithms that are not contained in the following list. A CryptoProvider implementation may not implement all algorithms that are contained in the following list.

Note: The string <HashFunction> in the following tables serves as a variable and can be replaced by any AlgId specified in the Table D.4 without the leading HASH/. E.g., "SYMKEY/HMAC/SHA-256" is a valid AlgId for an HMAC using the SHA-256 hash function.

AlgId	*Specification*
SYMKEY/AES-128	FIPS 197
SYMKEY/AES-192	FIPS 197
SYMKEY/AES-256	FIPS 197
SYMKEY/SM4	IETF draft-ribose-cfrg-sm4-10
SYMKEY/ChaCha	RFC 8439
SYMKEY/HMAC/<HashFunction>	FIPS 198-1

Table D.1: AlgIDs for SYMKEY

AlgId	*Specification*
PRIVKEY/RSA-2048	PKCS #1 V2.2
PRIVKEY/RSA-4096	PKCS #1 V2.2
PRIVKEY/Ed25519	RFC 8032
PRIVKEY/Ed448	RFC 8032
PRIVKEY/X25519	RFC 7748
PRIVKEY/X448	RFC 7748
PRIVKEY/ECDSA/secp256r1	ANSI X9.62
PRIVKEY/ECDSA/secp521r1	ANSI X9.62
PRIVKEY/ECDSA/brainpoolP256r1	ANSI X9.62, RFC 5639
PRIVKEY/ECDSA/brainpoolP512r1	ANSI X9.62, RFC 5639
PRIVKEY/ML-DSA-44	FIPS 204
PRIVKEY/ML-DSA-65	FIPS 204
PRIVKEY/ML-DSA-87	FIPS 204
PRIVKEY/ML-KEM-512	FIPS 203
PRIVKEY/ML-KEM-768	FIPS 203
PRIVKEY/ML-KEM-1024	FIPS 203

▽

△

PRIVKEY/SLH-DSA-SHA2-128f	FIPS 180-4, FIPS 205
PRIVKEY/SLH-DSA-SHA2-128s	FIPS 180-4, FIPS 205
PRIVKEY/SLH-DSA-SHA2-192f	FIPS 180-4, FIPS 205
PRIVKEY/SLH-DSA-SHA2-192s	FIPS 180-4, FIPS 205
PRIVKEY/SLH-DSA-SHA2-256f	FIPS 180-4, FIPS 205
PRIVKEY/SLH-DSA-SHA2-256s	FIPS 180-4, FIPS 205
PRIVKEY/SLH-DSA-SHAKE-128f	FIPS PUB 202, FIPS 205
PRIVKEY/SLH-DSA-SHAKE-128s	FIPS PUB 202, FIPS 205
PRIVKEY/SLH-DSA-SHAKE-192f	FIPS PUB 202, FIPS 205
PRIVKEY/SLH-DSA-SHAKE-192s	FIPS PUB 202, FIPS 205
PRIVKEY/SLH-DSA-SHAKE-256f	FIPS PUB 202, FIPS 205
PRIVKEY/SLH-DSA-SHAKE-256s	FIPS PUB 202, FIPS 205

Table D.2: AlgIDs for PRIVKEY

AlgId	*Specification*
PUBKEY/RSA-2048	PKCS #1 V2.2
PUBKEY/RSA-4096	PKCS #1 V2.2
PUBKEY/Ed25519	RFC 8032
PUBKEY/Ed448	RFC 8032
PUBKEY/X25519	RFC 7748
PUBKEY/X448	RFC 7748
PUBKEY/ECDSA/secp256r1	ANSI X9.62
PUBKEY/ECDSA/secp521r1	ANSI X9.62
PUBKEY/ECDSA/brainpoolP256r1	ANSI X9.62, RFC 5639
PUBKEY/ECDSA/brainpoolP512r1	ANSI X9.62, RFC 5639
PUBKEY/ML-DSA-44	FIPS 204
PUBKEY/ML-DSA-65	FIPS 204
PUBKEY/ML-DSA-87	FIPS 204

▽

△

PUBKEY/ML-KEM-512	FIPS 203
PUBKEY/ML-KEM-768	FIPS 203
PUBKEY/ML-KEM-1024	FIPS 203
PUBKEY/SLH-DSA-SHA2-128f	FIPS 180-4, FIPS 205
PUBKEY/SLH-DSA-SHA2-128s	FIPS 180-4, FIPS 205
PUBKEY/SLH-DSA-SHA2-192f	FIPS 180-4, FIPS 205
PUBKEY/SLH-DSA-SHA2-192s	FIPS 180-4, FIPS 205
PUBKEY/SLH-DSA-SHA2-256f	FIPS 180-4, FIPS 205
PUBKEY/SLH-DSA-SHA2-256s	FIPS 180-4, FIPS 205
PUBKEY/SLH-DSA-SHAKE-128f	FIPS PUB 202, FIPS 205
PUBKEY/SLH-DSA-SHAKE-128s	FIPS PUB 202, FIPS 205
PUBKEY/SLH-DSA-SHAKE-192f	FIPS PUB 202, FIPS 205
PUBKEY/SLH-DSA-SHAKE-192s	FIPS PUB 202, FIPS 205
PUBKEY/SLH-DSA-SHAKE-256f	FIPS PUB 202, FIPS 205
PUBKEY/SLH-DSA-SHAKE-256s	FIPS PUB 202, FIPS 205

Table D.3: AlgIDs for PUBKEY

AlgId	*Specification*
HASH/SHA-1	FIPS 180-4
HASH/SHA-224	FIPS 180-4
HASH/SHA-256	FIPS 180-4
HASH/SHA-384	FIPS 180-4
HASH/SHA-512	FIPS 180-4
HASH/SHA-512/224	FIPS 180-4
HASH/SHA-512/256	FIPS 180-4
HASH/SHA3-224	FIPS PUB 202
HASH/SHA3-256	FIPS PUB 202
HASH/SHA3-384	FIPS PUB 202
HASH/SHA3-512	FIPS PUB 202
HASH/SHAKE128	FIPS PUB 202
HASH/SHAKE256	FIPS PUB 202
HASH/SM3	

Table D.4: AlgIDs for HASH

AlgId	Specification
RNG/defaultRng	
RNG/TRNG	
RNG/Hash_DRBG/<HashFunction>	NIST SP 800-90Ar1
RNG/H-MAC_DRBG/<HashFunction>	FIPS 198-1, NIST SP 800-90Ar1
RNG/CTR_DRBG/AES-128	FIPS 197, NIST SP 800-38A, NIST SP 800-90Ar1
RNG/CTR_DRBG/AES-192	FIPS 197, NIST SP 800-38A, NIST SP 800-90Ar1
RNG/CTR_DRBG/AES-256	FIPS 197, NIST SP 800-38A, NIST SP 800-90Ar1

Table D.5: AlgIDs for RNG

AlgId	Specification
MAC/HMAC/<HashFunction>	FIPS 198-1
MAC/GMAC/AES-128	FIPS 197, NIST SP 800 38D
MAC/GMAC/AES-192	FIPS 197, NIST SP 800 38D
MAC/GMAC/AES-256	FIPS 197, NIST SP 800 38D
MAC/CMAC/AES-128	FIPS 197
MAC/CMAC/AES-192	FIPS 197
MAC/CMAC/AES-256	FIPS 197

Table D.6: AlgIDs for MAC

AlgId	Specification
SIG/SM2	IETF draft-shen-sm2-ecdsa-02
SIG/SLH-DSA-SHA2-128f	FIPS 180-4, FIPS 205
SIG/SLH-DSA-SHA2-128s	FIPS 180-4, FIPS 205
SIG/SLH-DSA-SHA2-192f	FIPS 180-4, FIPS 205
SIG/SLH-DSA-SHA2-192s	FIPS 180-4, FIPS 205
SIG/SLH-DSA-SHA2-256f	FIPS 180-4, FIPS 205
SIG/SLH-DSA-SHA2-256s	FIPS 180-4, FIPS 205
SIG/SLH-DSA-SHAKE-128f	FIPS PUB 202, FIPS 205
SIG/SLH-DSA-SHAKE-128s	FIPS PUB 202, FIPS 205
SIG/SLH-DSA-SHAKE-192f	FIPS PUB 202, FIPS 205
SIG/SLH-DSA-SHAKE-192s	FIPS PUB 202, FIPS 205
SIG/SLH-DSA-SHAKE-256f	FIPS PUB 202, FIPS 205
SIG/SLH-DSA-SHAKE-256s	FIPS PUB 202, FIPS 205
SIG/ML-DSA-44	FIPS 204
SIG/ML-DSA-65	FIPS 204
SIG/ML-DSA-87	FIPS 204
SIG/RSA-2048/PSS/<HashFunction>	PKCS #1 V2.2

▽

△

SIG/RSA-2048/PKCS1-v1_5/<HashFunction>	PKCS #1 V1.5
SIG/RSA-4096/PSS/<HashFunction>	PKCS #1 V2.2
SIG/RSA-4096/PKCS1-v1_5/<HashFunction>	PKCS #1 V1.5
SIG/Ed25519	RFC 8032
SIG/Ed25519ctx	RFC 8032
SIG/Ed25519ph/SHA-512	FIPS 180-4, RFC 8032
SIG/Ed25519ph/PRE-HASHED	RFC 8032
SIG/Ed448	RFC 8032
SIG/Ed448ph/SHAKE256	FIPS PUB 202, RFC 8032
SIG/Ed448ph/PRE-HASHED	RFC 8032
SIG/ECD-SA/secp256r1/<HashFunction>	ANSI X9.62
SIG/ECD-SA/secp521r1/<HashFunction>	ANSI X9.62
SIG/ECDSA/brain-poolP256r1/<HashFunction>	ANSI X9.62, RFC 5639
SIG/ECDSA/brain-poolP512r1/<HashFunction>	ANSI X9.62, RFC 5639

Table D.7: AlgIDs for SIG

AlgId	*Specification*
VER/SM2	IETF draft-shen-sm2-ecdsa-02
VER/SLH-DSA-SHA2-128f	FIPS 180-4, FIPS 205
VER/SLH-DSA-SHA2-128s	FIPS 180-4, FIPS 205
VER/SLH-DSA-SHA2-192f	FIPS 180-4, FIPS 205
VER/SLH-DSA-SHA2-192s	FIPS 180-4, FIPS 205
VER/SLH-DSA-SHA2-256f	FIPS 180-4, FIPS 205
VER/SLH-DSA-SHA2-256s	FIPS 180-4, FIPS 205
VER/SLH-DSA-SHAKE-128f	FIPS PUB 202, FIPS 205
VER/SLH-DSA-SHAKE-128s	FIPS PUB 202, FIPS 205
VER/SLH-DSA-SHAKE-192f	FIPS PUB 202, FIPS 205
VER/SLH-DSA-SHAKE-192s	FIPS PUB 202, FIPS 205
VER/SLH-DSA-SHAKE-256f	FIPS PUB 202, FIPS 205
VER/SLH-DSA-SHAKE-256s	FIPS PUB 202, FIPS 205
VER/ML-DSA-44	FIPS 204
VER/ML-DSA-65	FIPS 204
VER/ML-DSA-87	FIPS 204

▽

△

VER/RSA-2048/PSS/<HashFunction>	PKCS #1 V2.2
VER/RSA-2048/PKCS1-v1_5/<HashFunction>	PKCS #1 V1.5
VER/RSA-4096/PSS/<HashFunction>	PKCS #1 V2.2
VER/RSA-4096/PKCS1-v1_5/<HashFunction>	PKCS #1 V1.5
VER/Ed25519	RFC 8032
VER/Ed25519ctx	RFC 8032
VER/Ed25519ph/SHA-512	FIPS 180-4, RFC 8032
VER/Ed25519ph/PRE-HASHED	RFC 8032
VER/Ed448	RFC 8032
VER/Ed448ph/SHAKE256	FIPS PUB 202, RFC 8032
VER/Ed448ph/PRE-HASHED	RFC 8032
VER/ECD-SA/secp256r1/<HashFunction>	ANSI X9.62
VER/ECD-SA/secp521r1/<HashFunction>	ANSI X9.62
VER/ECDSA/brain-poolP256r1/<HashFunction>	ANSI X9.62, RFC 5639
VER/ECDSA/brain-poolP512r1/<HashFunction>	ANSI X9.62, RFC 5639

Table D.8: AlgIDs for VER

AlgId	*Specification*
SYMCIPHER/AES-128/CBC/PKCS7	FIPS 197, NIST SP 800-38A, PKCS #7
SYMCIPHER/AES-128/CBC/ISO9797-1	FIPS 197, NIST SP 800-38A, ISO/IEC 9797-1
SYMCIPHER/AES-128/CBC/PRE-PADDED	FIPS 197, NIST SP 800-38A
SYMCIPHER/AES-128/ECB/PKCS7	FIPS 197, NIST SP 800-38A, PKCS #7
SYMCIPHER/AES-128/ECB/ISO9797-1	FIPS 197, NIST SP 800-38A, ISO/IEC 9797-1
SYMCIPHER/AES-128/ECB/PRE-PADDED	FIPS 197, NIST SP 800-38A
SYMCIPHER/AES-192/CBC/PKCS7	FIPS 197, NIST SP 800-38A, PKCS #7
SYMCIPHER/AES-192/CBC/ISO9797-1	FIPS 197, NIST SP 800-38A, ISO/IEC 9797-1

▽

△

SYMCIPHER/AES-192/CBC/PRE-PADDED	FIPS 197, NIST SP 800-38A
SYMCIPHER/AES-192/ECB/PKCS7	FIPS 197, NIST SP 800-38A, PKCS #7
SYMCIPHER/AES-192/ECB/ISO9797-1	FIPS 197, NIST SP 800-38A, ISO/IEC 9797-1
SYMCIPHER/AES-192/ECB/PRE-PADDED	FIPS 197, NIST SP 800-38A
SYMCIPHER/AES-256/CBC/PKCS7	FIPS 197, NIST SP 800-38A, PKCS #7
SYMCIPHER/AES-256/CBC/ISO9797-1	FIPS 197, NIST SP 800-38A, ISO/IEC 9797-1
SYMCIPHER/AES-256/CBC/PRE-PADDED	FIPS 197, NIST SP 800-38A
SYMCIPHER/AES-256/ECB/PKCS7	FIPS 197, NIST SP 800-38A, PKCS #7
SYMCIPHER/AES-256/ECB/ISO9797-1	FIPS 197, NIST SP 800-38A, ISO/IEC 9797-1
SYMCIPHER/AES-256/ECB/PRE-PADDED	FIPS 197, NIST SP 800-38A
SYMCIPHER/SM4/CBC/PKCS7	IETF draft-ribose-cfrg-sm4-10, NIST SP 800-38A, PKCS #7
SYMCIPHER/SM4/CBC/ISO9797-1	IETF draft-ribose-cfrg-sm4-10, NIST SP 800-38A, ISO/IEC 9797-1
SYMCIPHER/SM4/CBC/PRE-PADDED	IETF draft-ribose-cfrg-sm4-10, NIST SP 800-38A
SYMCIPHER/SM4/ECB/PKCS7	IETF draft-ribose-cfrg-sm4-10, NIST SP 800-38A, PKCS #7
SYMCIPHER/SM4/ECB/ISO9797-1	IETF draft-ribose-cfrg-sm4-10, NIST SP 800-38A, ISO/IEC 9797-1
SYMCIPHER/SM4/ECB/PRE-PADDED	IETF draft-ribose-cfrg-sm4-10, NIST SP 800-38A
SYMCIPHER/ChaCha/8r	RFC 8439
SYMCIPHER/ChaCha/12r	RFC 8439
SYMCIPHER/ChaCha/20r	RFC 8439

Table D.9: AlgIDs for SYMCIPHER

AlgId	*Specification*
AEAD/AES-128/GCM	FIPS 197, NIST SP 800-38D
AEAD/AES-192/GCM	FIPS 197, NIST SP 800-38D
AEAD/AES-256/GCM	FIPS 197, NIST SP 800-38D
AEAD/SM4/GCM	IETF draft-ribose-cfrg-sm4-10, NIST SP 800-38D

▽

△

AEAD/ChaCha/8r/Poly1305	RFC 8439, RFC 8439
AEAD/ChaCha/12r/Poly1305	RFC 8439, RFC 8439
AEAD/ChaCha/20r/Poly1305	RFC 8439, RFC 8439

Table D.10: AlgIDs for AEAD

AlgId	*Specification*
ASYMENC/RSA-2048/OAEP/<HashFunction>	PKCS #1 V2.2
ASYMENC/RSA-4096/OAEP/<HashFunction>	PKCS #1 V2.2
ASYMENC/RSA-2048/ESAES-PKCS1-v1_5	FIPS 197, PKCS #1 V1.5, PKCS #1 V1.5
ASYMENC/RSA-4096/ESAES-PKCS1-v1_5	FIPS 197, PKCS #1 V1.5, PKCS #1 V1.5

Table D.11: AlgIDs for ASYMENC

AlgId	*Specification*
KDF/NistCounter/<HashFunction>	NIST SP 800-108
KDF/HKDF/<HashFunction>	RFC 5869
KDF/HKDF-1step/<HashFunction>	RFC 5869, RFC 5869
KDF/ANSI_X9_63/<HashFunction>	ANSI X9.63
KDF/PBKDF1/<HashFunction>	RFC 8018 / PKCS #5 v2.1
KDF/TLS12_PRF_MASTER_KEY_DERIVE/<HashFunction>	RFC 5246, PKCS #11 TLS 1.2 Mechanisms
KDF/TLS12_PRF_MASTER_KEY_DERIVE_DH/<HashFunction>	RFC 5246, PKCS #11 TLS 1.2 Mechanisms
KDF/TLS12_PRF_KEY_AND_MAC_DERIVE/<HashFunction>	RFC 5246, PKCS #11 TLS 1.2 Mechanisms
KDF/TLS12_PRF_KDF/<HashFunction>	RFC 5246, PKCS #11 TLS 1.2 Mechanisms

Table D.12: AlgIDs for KDF

AlgId	*Specification*
KEYWRAP/AESKEYWRAP/AES-128	FIPS 197
KEYWRAP/SHE/LOADKEY	AUTOSAR Specification of Secure Hardware Extensions

Table D.13: AlgIDs for KEYWRAP

AlgId	*Specification*
KEM/ML-KEM-512	FIPS 203
KEM/ML-KEM-768	FIPS 203
KEM/ML-KEM-1024	FIPS 203

Table D.14: AlgIDs for KEM

Key agreement is typically used in combination with a KDF. If a context includes the key derivation, it can be configured by concatenation of the AlgId used for key agreement and the AlgId used for key derivation. E.g., "KEYAGREEMENT/X25519/KDF/HKDF/SHA-256" is a valid AlgId for a context that performs an X25519 key agreement and a subsequent key derivation using HKDF with SHA-256.

AlgId	*Specification*
KEYAGREEMENT/X25519	RFC 7748
KEYAGREEMENT/X448	RFC 7748
KEYAGREEMENT/ECDH/secp256r1	ANSI X9.63
KEYAGREEMENT/ECDH/secp521r1	ANSI X9.63
KEYAGREEMENT/ECDH/brainpoolP256r1	RFC 5639, ANSI X9.63
KEYAGREEMENT/ECDH/brainpoolP512r1	RFC 5639, ANSI X9.63

Table D.15: AlgIDs for KEYAGREEMENT